PROVIDING AVAILABILITY IN THE LOWER LAYERS OF A WIRELESS NETWORK: PHILOSOPHIES AND CASE STUDIES

BY

JERRY T. CHIANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Doctoral Committee:

Assistant Professor Yih-Chun Hu, Chair
Professor P. R. Kumar
Professor Nitin Vaidya
Assistant Professor Nikita Borisov

ABSTRACT

Availability, together with confidentiality and integrity, form the basic concepts of information security. Indeed, a device, protocol, or network can enjoy the highest level of confidentiality and integrity, but still be of very little value to its clients if it is not available. In this dissertation, I consider the availability of wireless networks, which have permeated into almost every aspect of our daily life, from entertainment such as online games to utility infrastructure such as smart grids. A faulty entity can disrupt the connectivity offered by the lower layers of a wireless network and interrupt the provided services; thus, network availability in the lower layers is essential both to prevent inconveniences and to reliably provide critical services.

This dissertation considers three major philosophies that aim to provide availability by defending against denial-of-service (DoS) attacks at the lower layers. Specifically, a network can

1. detect misbehaviors to eliminate DoS attacks;

2. enforce fairness to mitigate DoS attacks; or

3. adopt randomization to escape DoS attacks.

For each philosophy, this dissertation presents a case study. I present a location verification protocol at the physical (PHY) layer that detects misbehavior in order to provide reliable position-based services. I also present a transport layer protocol that mitigates the flooding attack by enforcing Transport Control Protocal (TCP) fairness. I then present a flooding protocol that uses randomness at the PHY layer to escape from the jamming attack.

This dissertation is dedicated to my parents, whose support, encouragement, and love made this work possible.

ACKNOWLEDGMENTS

I remember reading once, "I thank the Lord, for there are too many people to whom I owe my gratitude." By no means can I write down a complete list of people I would like to thank, nor can I truly express my wholehearted gratitude to those I do list here.

I would like to thank the Lord, who gave me strength, so that I may finish what I started.

I would like to thank my parents, to whom I dedicate my dissertation. Their unconditional love and encouragement were both the motivation and the inspiration behind my work. I would like to thank Elaine, who has spent much time with me, and has given me laughter and joy over the past several years.

I would like to thank my adviser, Dr. Yih-Chun Hu, whose guidance helped me grew as a researcher. Over the past six years, Dr. Hu has been far more than just an adviser; he has been both a mentor and a friend, whose support I rely on – from tangible items such as conference travel to intangible items such as research philosophies.

I would like to thank all my friends for the many fruitful discussions over the years. Jason has helped me countless times, not only in technical fields but also in spiritual growth, and I would like to thank him greatly. Bernard has always been a good friend, who also encouraged me to finally graduate. Dongho and Jihyuk have provided me with many lively research discussions; their experimental implementations were included in this dissertation as well. I would like to thank the rest of my lab and my fellowship, who have always been there for me in times of need.

Finally, I would like to thank my committee for all the feedback and suggestions they have provided to make this a better dissertation.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Availability in the Lower Layers of a Wireless Network

A group of computing devices are *connected* if they can communicate with each other. Collaboratively, a group of connected computing devices can form a *computing network* that enables participants to share resources from data storage to computation ability. For example, users of a file-sharing network can share files among themselves; and clients of a computing grid can share idle computation cycles, memories, and storages. A single computing device may support multiple clients – known as end hosts – each of which performs a set of specific tasks. For example, a computer can simultaneously support multiple browser windows, each of which represents an Internet flow that can be considered a single end host. The ISO formally defines a common way that these computing devices and end hosts communicate with each other, known as the *Open System Interconnection* (OSI) model.

The OSI model presents a layered architecture in which each layer provides service to the layer above it. Starting from the bottom, the *physical (PHY) layer* handles the transmission and reception of raw bits over the communication channel. If the inter-device communication is done over a wired medium (such as telephone wire or optical cables), the resulting computing network is referred to as a *wired computing network*. Similarly, devices in a *wireless network* communicate with each other using the wireless medium.

The *data-link layer* resides above and collects raw bits from the PHY layer and delivers the collection of bits, known as a frame, to the layer above it. In other words, the data-link layer provides the functional and procedural means to transfer data between network entities. In particular, the data-link layer contains a sublayer known as *medium access control (MAC)*.

The MAC sublayer provides accessing and addressing functionalities so that multiple devices can share and communicate over a single communication channel. Above the data-link layer sits the *network layer* that provides routing functionality in a network based on the addressing service provided by the data-link layer. Above the network layer is the *transport layer* that provides end-to-end communication between applications on different devices. Together, the PHY, data-link and MAC, network, and transport layers are known as the *lower layers* of a network and provide connectivity between devices.

Above the lower layers are the session, presentation, and application layers; these three layers are collectively known as the *upper layers*. The upper layers use the connectivity provided by the lower layers to support user applications. These layers are not studied in this dissertation.

Instead, this dissertation focuses on the *availability* of the lower layers. Availability, along with confidentiality and integrity, form the basic concepts of *information security*. Indeed, a device, a protocol, or a network can enjoy the highest level of confidentiality and integrity, but is still useless to end users if it is not available.

While network applications can provide confidentiality and integrity by using end-to-end encryption and data correcting codes, a faulty entity can interrupt a computing network by disrupting the connectivity between devices. The use of networked devices has permeated into almost every aspect of our daily life, from entertainment such as online games to utility infrastructures such as a smart-grid. Consequently, network availability is essential both to prevent inconvenience and to provide uninterrupted critical services. Because the lower layers provide connectivity service to the higher layers, providing availability at the lower layers of a network is essential in keeping networks operational.

An attack that interrupts a service that is otherwise available is referred to as a *denial-of-service (DoS)* attack. In this dissertation, I consider three defense philosophies against DoS attacks targeting, and present case studies of applying these philosophies in, the lower layers of a wireless network:

1. Detect misbehaviors to eliminate DoS attacks;

2. Enforce fairness to mitigate DoS attacks; and

3. Adopt randomization to escape DoS attacks.

1.2 Eliminating Denial-of-Service by Detecting Misbehaviors

When the performance of a network suffers due to misbehaving insiders, we can detect and eliminate the insider attackers, thereby returning the network performance to normal. In Chapter 2, I present a novel location verification protocol to reject false location claims, thereby making position-based services available. For example, without the integrity of position claims, attackers can easily introduce wormholes into, and thus disrupt, networks that employ position-based routing. Part of the findings were published previously in [1].

1.3 Mitigating Denial-of-Service by Enforcing Fairness

However, in some cases, it is difficult to distinguish whether suboptimal network performance is caused by a malicious act or by under-provisioning the resource. Specifically, some events may cause abnormal network usage in time or location; we refer to these events as *flash events*.

For example, during the 2008 conflict between Georgia and Russia, many Georgian servers and websites were overloaded with request queries and brought offline. Experts suspect these websites were targeted by DoS attacks,[1] even though the same effect may occur if many people suddenly and simultaneously took an interest in the Georgian banking and political systems. In another example, when celebrity Michael Jackson passed away on June 25, 2009, there was an abnormal amount of queries about him and Google mistook the flash event for a DoS attack.[2] Some previous work seeks to distinguish flash events from malicious attacks using behavioral recognition [2]. While many different types of network resources can be exhausted, we specifically restrict our discussions to the network resource of *bandwidth* in Chapters 3 and 4.

[1] http://www.nytimes.com/2008/08/13/technology/13cyber.html

[2] googleblog.blogspot.com/2009/06/outpouring-of-searches-for-late-michael.html

To make bandwidth available, we do not necessarily need to make a distinction between flash crowd and DoS attackers. We can provide availability by enforcing fairness: as long as all network end hosts fairly share the communication medium, even if a portion of the network is experiencing congestion, each end host sharing that portion of network suffers, but not more than the rest of his peers. We adopt this approach to avoid the undesirable scenario of mistaking a flash crowd as malicious DoS attackers, in which case a portion of the flash crowd would experience self-induced unavailability.

There are many different notions of fairness. For example, in per-host bandwidth-fairness, each end-host can be granted an equal share of the total bandwidth; in per-host access-fairness, each end-host is granted an equal opportunity in accessing the communication medium; and in per-host time-fairness, each end-host is granted an equal time duration in accessing the communication medium. In this dissertation, I first study how an attacker can exploit the different notions of fairness so as to remain fair in one manner while denying availability in another manner; the findings were published in [3]. I then study how to enforce the de facto fairness of today's Internet to provide availability at the transport layer; part of the findings were published in [4].

1.4 Escaping Denial-of-Service by Adopting Randomization

When a network suspects there exist nearby misbehaving devices, yet wishes not to devote resources either to detect the misbehaviors or to enforce fairness among participants, the network can use randomization to opportunistically escape from the DoS attack. That is, the network repeatedly attempts to escape from DoS attacks. When successful, the network is able to bootstrap other security properties.

In Chapter 5, I study how to escape DoS attacks at the physical layer – jamming – by randomly orienting directional antennas to opportunistically eliminate interferences.

CHAPTER 2

LOCATION VERIFICATION: AN ESSENTIAL STEP IN SECURELY PROVIDING LOCATION-BASED SERVICES

In mobile and wireless networks, user location can be used to enhance network performance. For example, location information of a user has been incorporated in many position-based routing schemes in sensor networks [5, 6, 7] where power conservation is crucial. Learning the location of a receiver can also better aid a transmitter in orienting his directional antenna. However, if an attacker can successfully falsify his location claim in a network that uses geographic routing, that attacker can easily create wormholes in the network, and disrupt network connectivity by making the wormholes unusable. Therefore, to provide consistent service, all location claims should be securely verified before being used by a network.

Several approaches to secure location verification attempt to ensure that the user is within some circular region. These techniques generally assume a verifier is at the center of that region, and use physical limitations to prevent attackers outside of that region from successfully verifying the location. For example, Brands and Chaum [8] propose the distance bounding protocol that exploits the fact that attackers cannot send signals faster than the speed of light. In this chapter, I propose to build a protocol on top of Brands and Chaum's distance bounding protocol, but extend it to securely verify *a specific location* within the convex hull formed by the verifiers.

Throughout this chapter, I refer to the entity that claims and wishes to prove its location as the *prover*; and the entity that verifies such location claim, the *verifier*. To determine that a prover P is within distance ℓ from verifier V, the verifier can simply measure the round trip time. In particular, if the verifier sends a challenge to the prover, and the prover can correctly respond, so that the response reaches the verifier within time duration $2\ell/c$, then the prover (or an accomplice) must be located within a sphere centered at V with radius ℓ.

My protocol is based on the concept of *multilateration*: if a prover P proves to verifiers $V_1, \ldots, V_n$ that he is within radius ℓ_i from V_i, then *given there is only one prover*, he must be within the *intersection* of all n circles. Several previously proposed location verification protocols use distance bounding and multilateration. For example, Čapkun and Hubaux propose a multilateration scheme that specifies a δ test and a *point-in-triangle* test [9].

Naïve multilateration schemes, however, can be easily compromised: if the verifiers perform distance bounding independently at different times, then an attacker can move between the tests and prove his false location claim. Sastry et al. thus suggest the insight that a secure multilateration scheme must use *simultaneous* verification [10]. However, Chandran et al. show that simultaneous multilateration, while necessary to provide correctness in location verification, is insufficient to mitigate sophisticated collusion attacks [11].

In this chapter, I propose a simultaneous and intertwined verification protocol. The proposed verification protocol uses a novel challenge-response scheme to limit the time delay in challenge processing. Furthermore, the verifiers in the proposed protocol must be finely time-synchronized; prior studies suggest two verifiers can be synchronized to within 4% of the cable length between them. By neglecting the small delay in processing time and the synchronization error, I prove, without compromising the false alarm rate, that my protocol achieves the highest detection rate of false location claims achievable by any protocols based solely on time-of-flight measurements.

I further show that as long as the network can detect the distance enlargement attack, simultaneous multilateration is secure against the generic collusion attack. I propose a method that detects distance enlargement attack using *signal strength difference*.

2.1 Related Work

To verify a location claim that a prover is within a certain range from a verifier, Brands and Chaum propose the distance bounding protocol [8] in which the verifier rapidly exchanges challenges and responses with a prover using radio waves. If the verifier can receive the correct response to challenge $C(i)$ within time $t(i)$, the prover must, with high probability, reside *within*

a sphere of radius $\max_i \left(\frac{1}{2}ct(i)\right)$ around the verifier, where c is the speed of light.

In the *mafia fraud attack* [12], an attacker that is near the verifier acts as a man-in-the-middle between the verifier and a benign prover that is far away. Since it takes time for the attacker to forward a challenge to a benign user and subsequently forward the response to the verifier, the distance bounding protocol can effectively detect a mafia fraud attacker that claims to be closer to the verifier than the benign prover. However, a prover can arbitrarily delay his response so as to appear *farther* than he actually is, a misbehavior known as the *distance enlargement attack*. Brands and Chaum propose mitigating the distance enlargement attack by placing the verifier at the center of the region of interest. Consequently, enlarging the perceived distance does not benefit the prover.

Sastry et al. propose the Echo protocol [10] that uses an RF link for verifier-to-prover challenges and uses an ultrasonic link for prover-to-verifier responses. Thus if the prover responds within time $t(i)$ in the i^{th} round of the protocol, the prover must be at most ℓ away from the verifier, where s is the speed of sound and ℓ satisfies

$$\max_i \left(t(i)\right) = \frac{\ell}{c} + \frac{\ell}{s}.$$

Multilateration can be used to verify a precise location claim. Čapkun and Hubaux propose two tests for secure multilateration [9]. When the prover claims to be r away from a verifier V, and the perceived distance between the prover and a verifier is $\ell(i) = \frac{1}{2}ct(i)$ in the i^{th} round of the distance bounding protocol, the δ test limits the user's location ambiguity by rejecting the claim if

$$\max_i |r - \ell(i)| \geq \delta,$$

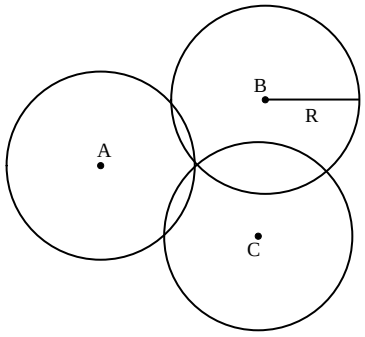
for some predefined δ. The *point-in-triangle* test ensures the claimed location is within the triangle formed by three verifiers.

While the δ test and the point-in-triangle test are sufficient to prevent false location claims from a single attacker, they cannot successfully defend against the *generic collusion attack* [11]. In the generic collusion attack, multiple attackers collaborate in an effort to deceive the verifiers into accepting an incorrect location claim.

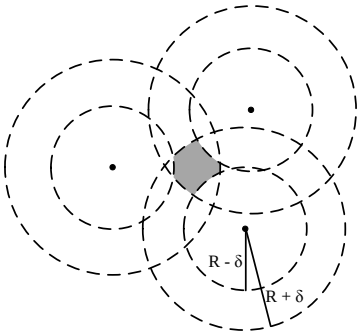
For example, in Figure 2.1(a), the prover claims to be located at a distance R away from all three verifiers. However, due to processing delay and error in the location estimation, the prover may appear to be slightly farther away from each of the verifiers. The verifiers thus allow a time uncertainty of up to $2\delta/c$, or a spatial uncertainty of δ. In Figure 2.1(b), the dashed lines denote the region that is at most $R \pm \delta$ away from each verifier. If the prover has no processing delay, he must be located in the shaded region in the figure in order to prove his location claim. In Figure 2.1(c), three colluding attackers who share their cryptographic identities can each pass the δ test made by one verifier. The attackers, after passing the δ tests, will also pass the point-in-triangle test since the claimed location is indeed in the interior of the triangle formed by the three verifiers. The verifiers then mistakenly accept the false location claim of the colluding attackers. Sastry et al. thus observe that multilateration must be done in a manner such that all verifications are performed simultaneously [10].

Several studies propose collusion-resilient verification schemes that do not solely rely on time-of-flight measurements. If the provers can be uniquely identified, then collusion, where many provers pretend to share a single identity, is not possible. Čapkun and Hubaux suggest two schemes to mitigate collusion attacks [13]. In the first scheme, the verifiers fingerprint all users, thereby asserting that different users cannot act as one single entity. In the second scheme, the provers are all given tamper-proof hardware; thus the attackers cannot clone a node and subsequently collude. The first approach requires additional assumptions about the system's tolerance of the variability of fingerprints and also the attackers' ability to tune his own fingerprints. The second approach, on the other hand, can be very expensive if special hardware needs to be installed on a large number of provers. The approach of using tamper-proof hardware to defend against colluding attackers was independently proposed also by Singelée and Preneel [14]. These two approaches are orthogonal to my proposed protocol and can be incorporated to provide additional security features.

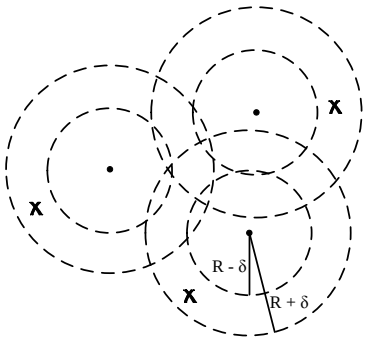
Čapkun et al. propose a location verification protocol where some verifiers remain hidden and mobile [15]. Since the attackers are not aware of these hidden verifiers, a forged signal sent by the attackers will not reach these hidden verifiers at the correct time or incident angle, depending on whether the underlying system uses time-of-flight measurements or angle-of-arrival



(a) Prover is R away from all three verifiers



(b) Acceptance zone



(c) Colluding attackers

Figure 2.1: Illustration of the δ test

measurements. Such an approach is secure if and only if the hidden mobile verifiers can actually be made completely hidden. Chaudran et al. note an attack scheme that uses trial and error to find the location of the hidden verifiers if their mobility is limited [11].

Chandran et al. propose an attack algorithm that shows that any location verification schemes based solely on time-of-flight measurements must be susceptible to the generic collusion attack [11]. In this chapter, I show that the collusion attacks can be defeated by preventing distance enlargement attacks. In particular, I propose using the difference in signal strength to detect distance enlargement attacks. Cai et al. independently propose a similar approach in detecting and pairing close-by devices [16].

2.2 Attacker Model and System Assumptions

2.2.1 Attacker Model

In this chapter, I only consider attackers that possibly collude to deceive the verifiers about their locations. Specifically, I do not consider attackers that seek to disrupt the challenge-response channels in the verification system. I also make no restriction on the information the attackers share among themselves.

Čapkun and Hubaux gave an in-depth analysis and concluded that RF time-of-flight based verification systems exhibit the best security properties compared to other techniques, such as angle-of-arrival and ultrasound time-of-flight techniques [9]. Since there is no known working communication technique that is faster than the speed of light, I assume an attacker cannot communicate with the set of verifiers or other colluding attackers faster than the speed of light.

The round trip time-of-flight of an RF signal is thus at least $\frac{2\ell}{c}$, where ℓ is the physical length of the line-of-sight signal path between the prover and the verifier. We calculate the *perceived distance* as $\lambda = \frac{1}{2}ct$, where t is the measured round trip time-of-flight; a prover cannot decrease his perceived distance beyond the length of the line-of-sight path from a verifier, but can enlarge his perceived distance by delaying the response or sending the signal along a longer path. The verifier can safely assume that the prover is located

no farther away than the perceived distance. I assume the attackers are able to instantaneously respond to challenges since it is difficult to determine any meaningful lower bound in processing time.

Finally, and only for my optimized scheme that uses signal strength measurements, I also assume that the attackers are sufficiently far away from each verifier so that the attackers are in the far-field region of every verifier's antennas.

2.2.2 System Assumptions

While my proposed protocol makes very few assumptions about the attackers, it requires that all verifiers be trustworthy, secure, and able to weakly time synchronize among themselves. My proposed protocol also assumes that each verifier V_i knows its own location loc_{V_i}.

I assume that each verifier can communicate with every other verifier using a separate secure communication channel. These assumptions can be achieved by having physically secure verifiers communicate over secure wired links. I also assume that each prover shares a secret key with all verifiers.

The granularity to which verifiers can synchronize time among themselves directly affects the accuracy of the location proof. To effectively synchronize time, verifiers can synchronize over wires directly connecting them to each other. Vook et al. show that using a crossover cable 1 m in length, two HP5372A frequency and time interval analyzers can time synchronize and filter the machine jitters so the standard deviation of the synchronized time is 0.771 ns, equivalent to a spread of 2.5 ns with 99.7% confidence [17]. Ishikawa and Mita show that sensors connected using a 190 m LAN cable can time synchronize to within 25 ns, equivalent to a distance uncertainty of 4% of the cable length [18].

My optimized protocol uses signal strength measurements and makes the following radio model assumptions. First, an antenna's far-field distance is given by

$$\ell_{ff} = \frac{2G^2}{\text{signal wavelength}},$$

where G is the maximum dimension of the antenna. Second, the receiver's system noise temperature T_{sys} is less than the typical noise temperature of the quiet Sun, which is around 50,000 K at 2.4 GHz. For example, let a

Table 2.1: List of symbols and definitions

Symbol	Definition
V_i	The i^{th} verifier
P	The prover
loc_A	Location of A
$\widehat{\mathsf{loc}_A}$	Estimated or claimed location of A
r_i	The *claimed* distance between P and V_i
ℓ_i	The line-of-sight distance between P and V_i
ℓ_i'	The distance of the signal path between P and V_i
u_i	The uncertainty measured by V_i
δ_i	Acceptance threshold of V_i
γ	Path loss exponent
d	Separation between antennas
λ_i	The *perceived* distance between P and V_i

transmitter transmit a 2.4 GHz signal with bandwidth $\Delta f = 1$ MHz using an SMCANT-DI145 directional antenna,[1] which has length 0.316 m. Let the receiver also receive using an SMCANT-DI145 directional antenna; then the far-field distance of the SMC antenna is at $\ell_{ff} = 1.66$ m and the system noise is given by

$$N_{sys} = kT_{sys}\Delta f = -96.61 \text{ dbm},$$

where k is the Boltzmann constant. Finally, I use a log-distance path-loss model. Since I assume the attackers are in the far-field, I do not consider the signal behavior in the near-field, wherein the signal strength must be capped.

2.3 Simultaneous Multilateration

Since my proposed protocol uses only radio waves, I will normalize the distance and time with respect to the speed of light throughout the remainder of this chapter. That is, *a unit distance is defined to be the distance traveled by radio wave over one unit time.* Table 2.1 gives a list of symbols and definitions.

[1]Data sheet available at: http://www.smc.com/files/AP/DS_ANT.pdf

2.3.1 The Simultaneous Multilateration Protocol

A secure multilateration scheme must be performed *simultaneously* by modifying the distance bounding protocol so that the prover simultaneously responds to the challenge from each verifier. If there are a total of N verifiers, then each verifier sends a separate challenge, and the prover proves that he has received all the challenges by combining all N challenges using a mathematical function. This function should have the property that when given N inputs, it produces a deterministic output; however, when given $M < N$ inputs, all outputs are equally likely.

My protocol can use any modulation scheme and multiple access protocol so long as they provide the decoding speed required. However, for the simplicity of describing my protocol, I adapt frequency shift keying for bit transmission, and frequency division to allow multiple senders to send simultaneously. That is, a verifier V_i is allocated two frequencies, f_{i0} and f_{i1}. To transmit the bit 0, V_i transmits a single tone on frequency f_{i0}, and to transmit the bit 1, V_i transmits a single tone on frequency f_{i1}. If the prover detects a signal on f_{i0} and not on f_{i1}, the prover decodes a 0 from V_i. If the prover detects a signal on f_{i1} and not on f_{i0}, the prover decodes a 1 from V_i. Otherwise the prover makes no decision.

Rasmussen and Čapkun recently proposed using *challenge reflection with channel selection (CRCS)* to realize the distance bounding protocol [19]. In CRCS, the single distance bounding verifier selects a frequency channel on which to send his challenge; the prover then responds by mixing the challenge with another sinusoid, whose frequency depends on a bit stream B agreed between the prover and the verifiers. The verifier then verifies that the offset between challenge and response frequencies corresponds to the correct response. The authors show that the prover is able to receive, turn around, and respond within less than 1 ns. I propose extending the CRCS protocol for my simultaneous multilateration protocol, which I refer to as the sim-CRCS protocol. In sim-CRCS, each verifier selects a different frequency; the prover then responds by mixing all received challenges and a sinusoid based on the agreed bit stream B. The processing time experienced by the prover should be similar to that shown by Rasmussen and Čapkun since a mixer can mix multiple inputs at once.

In my protocol, the prover P initiates the verification request by first securely submitting his claimed location $\widehat{\mathsf{loc}_P}$ using his shared secret with the verifiers. All N verifiers then time synchronize among themselves using their own secure channel. Each verifier V_i then chooses a challenge bit C_i and a unique frequency f_{iC_i}, and then informs other verifiers of his choice. All verifiers then collectively decide an arrival time τ, and each verifier calculates the transmission time to transmit his challenge by subtracting from the agreed arrival time the propagation time between the verifier and the prover. That is,

$$\text{Transmission time of } V_i = \tau - \mid \mathsf{loc}_{V_i} - \widehat{\mathsf{loc}_P} \mid = \tau - r_i,$$

where r_i is the claimed distance between prover and verifier V_i. Obviously, practicality dictates that the transmission time be after the current time, and this translates into a requirement for agreeing on τ. Each verifier finally transmits a tone on f_{iC_i} at its transmission time so that all N challenges *arrive at the claimed location simultaneously at* τ.

If the location claim is correct (i.e., $\widehat{\mathsf{loc}_P} = \mathsf{loc}_P$), the prover receives all challenges simultaneously at time τ, determines the responding frequency by sim-CRCS, and then responds by broadcasting a tone on the correct frequency, which is in turn received by all verifiers. For example, a prover can receive all challenges from different bands using an ultra-wide-band antenna; the prover then feeds the signal consisting of all challenges into a diode to mix all challenges. This is equivalent to one round of bit exchange in the original distance bounding protocol [8]. To perform another round of my protocol, each verifier selects a fresh set of challenge frequencies and repeats.

When each verifier receives the response from the prover, the verifier first checks if the response bit is correct. If the response bit value is incorrect, the location claim is rejected. If the verifiers do not receive a response or if the verifiers receive an ambiguous response (i.e., receiving both 0 and 1), then without penalizing the prover, the verifiers will initiate the next round. If the response bit value is correct, each verifier checks the elapsed time between when the verifier sent his challenge and when the response was received. Since the response bit is only one bit in length, the prover has a 50% chance to reply correctly by simply guessing. As in the basic distance bounding protocols, my protocol is run many rounds where a portion of responses must be correct in order to exponentially diminish the probability that the prover

guesses correctly every round. After each verifier has calculated the region in which the prover must reside, the verifiers can intersect these regions to verify the location claim.

2.3.2 Uncertainty: Quantitatively Studying the Security of Timing-Based Location Verification Protocols

Let each verifier calculate the round trip time by taking the difference between the time when it sent the challenge and the time when it received the response. If the claimed location is at a distance of r_i from verifier V_i, and the correct response is not received until $2\ell_i$ after the challenge was sent, then define the *uncertainty* to be $u_i = 2\ell_i - 2r_i$. The amount of uncertainty a verifier V_i accepts is given by a threshold δ_i, which can vary based on the claimed location, the purpose of the location proof, and other factors. Once each δ_i is determined, the location claim is accepted if

$$|u_i| \leq \delta_i \quad \forall V_i.$$

In location verification systems that use only time-of-flight measurements, a sophisticated attacker can only be caught falsifying a location claim if he cannot correctly respond within some allowed elapsed time. That is, a sophisticated attacker can only be caught cheating if the uncertainty measured by any verifier is *greater than* the threshold, since an attacker can use directional antennas to inject verifier-specific delay when any measured uncertainty is negative. In other words, the uncertainty (not its absolute value) can be considered a measure of the level of security provided by a location verification protocol.

To analyze my proposed simultaneous multilateration protocol, let there be a set of N verifiers $\mathbb{V} = \{V_1, \ldots V_N\}$ and a prover P. Let the *line-of-sight distance* and the *claimed distance* between prover P and verifier V_i be ℓ_i and r_i, respectively. In my protocol, each verifier V_i sends his challenge at time $-r_i$ so that all N challenges reach the claimed location at time 0. However, since the prover is actually ℓ_i away, the prover cannot collect all challenges until time $\max_{V_n \in \mathbb{V}} (\ell_n - r_n)$. The prover then spends o_i time to process the challenges and respond to all verifiers. The response would take ℓ_i to travel from P to verifier V_i for a total measured time-of-flight of

$\ell_i + \max_{V_n \in \mathbb{V}} (\ell_n - r_n) + r_i + o_i$. The corresponding uncertainty measured by verifier V_i is $u_i = \ell_i - r_i + \max_{V_n \in \mathbb{V}} (\ell_n - r_n) + o_i$. As outlined in my attacker model, my analyses in Sections 2.4 and 2.5 assume $o_i \to 0$.

2.4 Optimality of Simultaneous Multilateration

To show the optimality of my proposed protocol, I assume an ideal environment where both the processing delay o_i and the synchronization errors are negligible. By observing the incurred uncertainty, I can show that, without compromising the false alarm rate, my protocol achieves the highest detection rate of false location claims that any location verification schemes based solely on time-of-flight can provide.

I first note that a correct location claim is trivially accepted since the measured uncertainty is negligible ($o_i \to 0$). I then show that an incorrect location claim incurs the maximum uncertainty in my protocol. Equivalently, if a false location claim can be detected by any other verification protocols based solely on time-of-flight measurements, that false claim can also be detected by my protocol.

My analysis is based on the real uncertainty, and not its absolute value. This is because any sophisticated attacker can always delay his response to compensate for negative uncertainty. In other words, the ability to reject claims based on negative uncertainty does not provide any security benefit independent of assumptions on attacker capability, and is thus not considered in my analysis.

Theorem 2.4.1. *The simultaneous multilateration protocol described in Section 2.3.1, without compromising the false alarm rate, provides the highest detection rate of false location claims that can be provided by any protocols based solely on time-of-flight measurements. That is, with a given topology of verifiers and provers, if a set of colluding attackers can deceive a particular verifier in the proposed verification system, they would be able to deceive the same verifier in all other verification systems based on time-of-flight measurements alone.*

Proof. I prove my theorem by showing that with a sophisticated attacker, the uncertainty measured by any verifier in my system is an *upper bound*

of the uncertainty measured by that same verifier in any systems based on time-of-flight alone.

Let the collection of verifiers be $\mathbb{V}$, and the set of challenge-transmitting verifiers be $\mathbb{V}_t \subseteq \mathbb{V}$. I assume that the locations of all verifiers, transmitting or silent, are known to the public since I am only interested in comparing time-of-flight based verification systems. Let there be a set of provers $\mathbb{P}$, and prover $P_k \in \mathbb{P}$ is ℓ_{ki} away from $V_i \in \mathbb{V}$. The set of provers collaboratively seek to prove a single location claim that is r_i away from verifier V_i. Finally, let verifier $V_i \in \mathbb{V}_t$ transmit his challenge at time t_i.

Without loss of generality, I assume that the challenges generated by the set of transmitting verifiers $\mathbb{V}_t$ are intertwined. Meaning, the correct prover response is dependent on all challenges sent by transmitting verifiers. If only subsets of challenges are intertwined, then each corresponding subset of transmitters can be considered as the set of transmitting verifiers and the rest are considered as silent verifiers. Thus, a set of verifiers can be regrouped into several sets with this observation. That is, any system that does not intertwine all its challenges can be considered a set of systems, not necessarily mutually exclusive, each intertwining all its challenges. The rest of the proof shows that the uncertainty measured by each system, and thus also the maximum measurement, is less than or equal to that measured by the proposed verification protocol.

Observe that the challenge from V_i reaches the claimed location at $t_i + r_i$. Since the challenges are intertwined, V_i expects the prover to respond at time $\max_{V_i \in \mathbb{V}_t} (t_i + r_i)$ after all challenges reach the prover. Without considering the processing time, o_i, the response then reaches V_i at time

$$E_i = \max_{V_n \in \mathbb{V}_t} (t_n + r_n) + r_i.$$

The same challenge from V_i would reach P_k at $t_i + \ell_{ki}$. Hence, the earliest response from a prover can reach verifier V_i at time

$$\min_{P_k \in \mathbb{P}} \left(\max_{V_n \in \mathbb{V}_t} (t_n + \ell_{kn}) + \ell_{ki} \right).$$

The corresponding uncertainty ϕ_i, measured by V_i, is simply the difference:

$$\phi_i = \min_{P_k \in \mathbb{P}} \left(\max_{V_n \in \mathbb{V}_t} (t_n + \ell_{kn}) + \ell_{ki} \right) - E_i.$$

In the simultaneous multilateration system, all verifiers send challenges that are intertwined, and the smallest uncertainty measured by verifier V_i is

$$u_i = \min_{P_k \in \mathbb{P}} \left(\max_{V_n \in \mathbb{V}} (\ell_{kn} - r_n) + \ell_{ki} - r_i \right).$$

Adding and subtracting $\max_{V_n \in \mathbb{V}_t} (t_n + r_n)$:

$$u_i = \min_{P_k \in \mathbb{P}} \left(\max_{V_n \in \mathbb{V}} (\ell_{kn} - r_n) + \ell_{ki} + \max_{V_n \in \mathbb{V}_t} (t_n + r_n) \right) - E_i.$$

The minimization of u_i is done over the set of provers, and is independent of the maximization inside, performed over the set of verifiers $\mathbb{V}$. Therefore, by changing the maximization to be performed over the subset of transmitting verifiers $\mathbb{V}_t$, the uncertainty must decrease or remain the same:

$$u_i \geq \min_{P_k \in \mathbb{P}} \left(\max_{V_n \in \mathbb{V}_t} (\ell_{kn} - r_n) + \ell_{ki} + \max_{V_n \in \mathbb{V}_t} (t_n + r_n) \right) - E_i.$$

Since the sum of the maxima is larger than the maximum of the sum, collapse the two maximum terms inside:

$$\begin{aligned} u_i &\geq \min_{P_k \in \mathbb{P}} \left(\max_{V_n \in \mathbb{V}_t} (\ell_{kn} - r_n + t_n + r_n) + \ell_{ki} \right) - E_i \\ &= \min_{P_k \in \mathbb{P}} \left(\max_{V_n \in \mathbb{V}_t} (\ell_{kn} + t_n) + \ell_{ki} \right) - E_i \\ &= \phi_i. \end{aligned}$$

■

2.5 Collusion Attack against Simultaneous Multilateration

When a single prover makes a location claim, my intertwined verification ensures that he is where he claimed because he must perform all the verifications simultaneously, and his response would be late to some verifiers if he made a false location claim. However, while a single attacker cannot cheat all the verifiers, he may be able to cheat a subset of verifiers. For example, imagine three verifiers V_1, V_2, V_3 forming a regular triangle, an attacker sit-

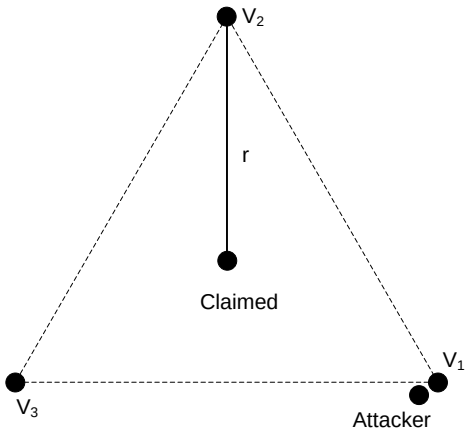


Figure 2.2: Illustration of an attacker that is near one of verifiers making a false location claim

ting next to verifier V_1, and a claimed location at the geometric center of the triangle as shown in Figure 2.2. Let the distance between the verifiers and the center be r. All three verifiers would send challenges at time $-r$ and expect the correct response at time r. The attacker receives a challenge from V_1 at time $-r$, and challenges from V_2, V_3 at time $\left(\sqrt{3}-1\right)r < r$. Thus, by delaying his response and transmitting using a directional antenna, the attacker is able to respond to only verifier V_1 at the expected time r.

Since one attacker may attack a subset of verifiers, a set of attackers may be able to collude and attack the entire system in a distributed manner. In the rest of this section, I give a mathematical model that allows us to completely characterize the feasibility of collusion attacks.

From Section 2.3.2, I observe that the uncertainty measured by verifier V_i is

$$u_i = (\ell_i - r_i) + \max_n (\ell_n - r_n),$$

where ℓ_i is the perceived distance from the prover to verifier V_i and r_i is the distance from the claimed location to verifier V_i. An attacker can deceive verifier V_i if $u_i \leq 0$. In the example given above,

$$\max_n (\ell_n - r_n) = \max_n \ell_n - r = (\sqrt{3}-1)r$$

and

$$\ell_1 - r_1 = -r_1 = -r$$

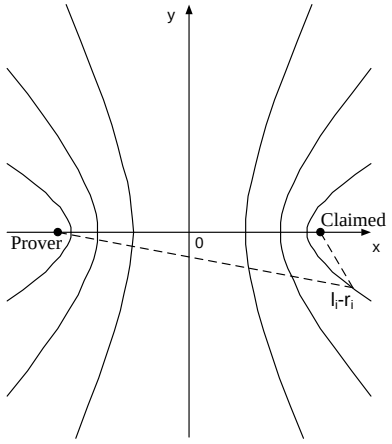


Figure 2.3: Hyperbolic contour of the difference in distances

for a sum that is less than zero.

To completely characterize my system, one needs to analyze the quantity $\ell_i - r_i$, the difference in the line-of-sight distance from attacker to verifier V_i, and the distance from the claimed location to V_i. It is known that a hyperbola with foci f_1 and f_2 has the property that the differences in distances from any points on the hyperbola to the foci have the same magnitude. Therefore, let the attacker and the claimed location be the foci; then the contour of the quantity $\ell_i - r_i$ is simply a collection of hyperbolas. This section analyzes two special cases: the first scenario has three verifiers forming a triangle, and the second scenario has infinitely many verifiers densely distributed on the boundary of a convex space. In both cases, the claimed location is assumed to be inside the convex hull. These two cases present the two extremes in number of verifiers used in a location verification system.

To analyze both cases, I first orient the prover and the claimed location so that the prover is at $-d$ and the claimed location is at $+d$ on the x-axis as shown in Figure 2.3. I will refer to the contour that is perpendicular to the x-axis as the y-axis; this contour presents the collection of points that are equidistant from the prover and from the claimed location. Each hyperbola is made up of two contours that are symmetric about the y-axis. The two contours have the same magnitude but opposite signs. In particular, the contour to the right of the y-axis (closer to the claimed location) represents

a positive value of $\ell_i - r_i = a > 0$, and, similarly, the contour to the left of the y-axis (closer to the prover) represents a negative value of $\ell_i - r_i = -a < 0$.

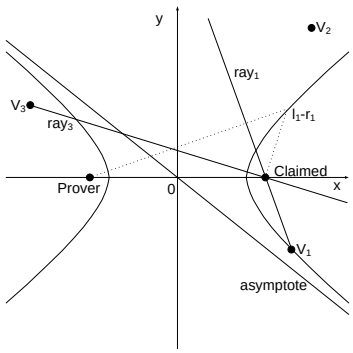
The first scenario is illustrated in Figure 2.4(a). Since the claimed location is in the interior of the verifier triangle, one of the verifiers, V_1, must be located to the right of the claimed location, on the contour $\ell_1 - r_1$. Furthermore, $\ell_1 - r_1 > 0$ since it lies on a contour closer to the claimed location than the prover. Since the contour plot is also symmetric about the x-axis, without loss of generality, let the verifier V_1 have a negative y value. Then draw the asymptote of this particular hyperbolic contour running through quadrants II and IV, as shown in Figure 2.4(a).

A contour that represents a value less than the opposite of the $\ell_1 - r_1$ contour must be entirely to the other side of the hyperbolic asymptote. In other words, an attacker can deceive a verifier only if that verifier is located to the left of the hyperbolic asymptote. Let verifier V_3 be vulnerable to attack because it measures a negative uncertainty; that is, V_3 is located to the left of the asymptote. Now draw a ray from verifier V_1 through the claimed location, and another ray from verifier V_3 through the claimed location. The two rays are named ray_1 and ray_3, respectively, as shown in Figure 2.4(a). The two rays intersect at the claimed location, and the other verifier V_2 must be located above[2] both rays in order to enclose the claimed location. However, ray_1 never intersects the asymptote because it is steeper than the asymptote; ray_3 intersects the asymptote once, but such intersection must be between V_3 and the claimed location. Ergo, V_2 can only be located on the right side of the asymptote, and $\ell_2 - r_2 > -(\ell_1 - r_1)$.

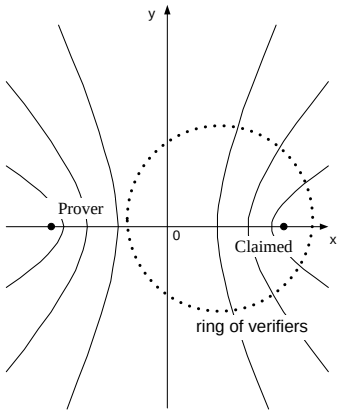
Consequently, if there are three verifiers enclosing the claimed location, then an attacker can cheat at most one verifier (e.g. V_3), by himself; thus regardless of where the attacker is, he needs at least two other colluding attackers to deceive the simultaneous multilateration system with three verifiers.

In the second scenario, let there be infinitely many verifiers densely distributed on the *boundary of a convex space*. Since the verifiers are densely distributed, there must be at least one verifier located on the ray from the claimed location to positive infinity, call it V_{pos}, and observe $\ell_{pos} - r_{pos} = 2d$.

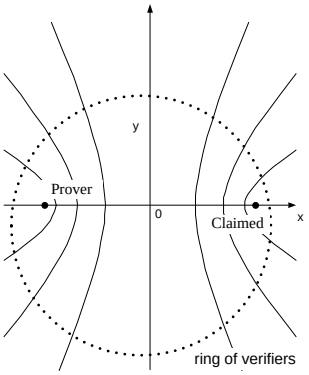
[2] V_2 is located "above" the rays because the segment connecting V_1 and V_3 is below the claimed location. Had I chosen V_1 with positive y value, V_2 would be located "below" the rays.



(a) Three verifiers



(b) Infinitely many verifiers, attacker outside the convex space



(c) Infinitely many verifiers, attacker inside the convex space

Figure 2.4: Illustrations of the scenarios studied in Section 2.5

If the prover is located outside the convex space, as shown in Figure 2.4(b), then no verifiers are located on the $-2d$ contour, which is a ray from the prover to negative infinity. Consequently, no verifier can be cheated by the attacker. If the prover is otherwise located inside the convex space, as shown in Figure 2.4(c), then only the verifiers located on the ray from the prover to negative infinity can be cheated. Since an attacker is needed in every orientation to deceive the verifiers, the attacker needs the same order of colluders as there are verifiers in the system.

Because all convex hulls can be partitioned into triangles, if the claimed location is inside the convex hull, it must be inside at least one triangle. I thus establish a very loose bound that attackers need at least two other colluders to deceive a simultaneous multilateration system with three or more verifiers.

2.6 Mitigating the Distance Enlargement Attack

In this section, I explore a possible optimization that can detect collusion attacks, and reject false claims from colluding attackers. In Section 2.3.2, I show that the uncertainty measured by verifier V_i is

$$u_i = \ell_i - r_i + \max_n (\ell_n - r_n) .$$

If there exists a method so that the length of signal path between a prover and a verifier can be accurately measured, then the verifier could enforce that the signal travels a distance consistent with the claimed distance: $\ell_i \approx r_i$. The measured uncertainty then increases to

$$u_i \approx \max_n (\ell_n - r_n) \geq 0.$$

Hence, u_i is close to 0 *only if* the claimed location is correct.

Since the distance bounding protocol already provides an accurate *upper bound* on the distance between a prover and a verifier, *eliminating the distance enlargement attack is sufficient to also eliminate the generic collusion attack when using simultaneous multilateration.* I propose using *signal strength difference* to obtain a lower bound on the distance between a prover and a verifier. Even though Cai et al. propose using a similar approach to verify nearby neighbors [16], this is a first attempt to mitigate the dis-

tance enlargement attack, which has been generally accepted as an inherent weakness in secure location verification schemes [9].

In my proposed optimization, each verifier is equipped with two highly-directive antennas that are placed so the orientations are almost collinear without shadowing each other. After receiving a location claim, each verifier orients both its direction antennas toward the claimed location. Each verifier then uses each of his antennas to measure the signal strength of the prover's response. Let the distance between the antennas be d, the line-of-sight distance and the length of the actual signal path from the prover to the closer antenna be ℓ and ℓ', respectively. The length-of-signal-path of a non-line-of-sight signal can be longer than the physical distance between the prover and the verifier. The resulting measured uncertainty is $u = \ell' - r + \max_n (\ell_n - r_n)$.

Subtract the measured signal strength of the farther antenna from that of the closer antenna, and obtain

$$\Delta s(\ell', \gamma) = 10\gamma \log \left(\frac{\ell' + d}{\ell'} \right) \geq 0 \text{ dB},$$

where γ is called the *path loss exponent*.

If the path loss exponent is consistent over time and space, then the difference in signal strength can provide an accurate distance measurement. However, because the path loss exponent varies with respect to time and space, we must analyze the extent the path loss exponent can be used to mitigate the generic collusion attack. In particular: if the length of the signal path between a prover and a verifier is ℓ', but the prover claims to be $r > \ell'$ away from the verifier, what is the threshold $\frac{\ell'}{r} < \eta$ such that a verifier can detect the distance enlargement attack and reject the location claim?

Since the two verifier antennas and the prover are collinear, the paths from the prover to the two directional antennas are very similar and should result in similar path loss exponents. In order for a response to be accepted, its measured signal strength difference must be larger than that induced by the claimed distance and *minimum* path loss exponent $\Delta s(r, \gamma_{\min})$ and smaller than that induced by the claimed distance the *maximum* path loss exponent $\Delta s(r, \gamma_{\max})$:

$$10\gamma_{\min} \log \left(1 + \frac{d}{r} \right) \leq 10\gamma \log \left(1 + \frac{d}{\ell'} \right) \leq 10\gamma_{\max} \log \left(1 + \frac{d}{r} \right).$$

Simplifying the inequality on the right-hand side:

$$\ell' \geq \frac{d}{\left(\exp\left(\frac{\gamma_{\max}}{\gamma}\ln\left(1+\frac{d}{r}\right)\right)-1\right)}.$$

If the term $\frac{d}{r}$ is small, then the above expression can be approximated using first-order Taylor expansion: $\frac{\ell'}{r} \geq \frac{\gamma}{\gamma_{\max}} \geq \frac{\gamma_{\min}}{\gamma_{\max}}$, or $\min \ell' = r\frac{\gamma_{\min}}{\gamma_{\max}}$.

This requirement in measured distance enables us to enforce that the length of signal path between a prover and a verifier is at least a constant factor of the prover's claim. For example, if I consider $\gamma_{\min} = 2$ and $\gamma_{\max} = 4$, then a verifier using signal strength difference can enforce that the distance between itself and a prover is at least half that claimed. This property improves the uncertainty measured by verifier V_i to

$$u_i = \max\left\{\ell_i, r_i\frac{\gamma_{\min}}{\gamma_{\max}}\right\} - r_i + \max_{V_n \in \mathbb{V}}(\ell_n - r_n).$$

2.7 Security Analysis of the Optimized Scheme

Instead of the more cost-effective choice of two omni-directional antennas, I choose to equip each verifier with two highly-directive antennas in order to take advantage of two desirable properties:

1. Directionality enables both antennas to hear the same response; and
2. Directivity alleviates the impact of fading by rejecting signals from secondary paths.

A directional antenna does not suffer as much from multipath as an omni-directional antenna would; thus using highly-directive antennas in my protocol also mitigates fading and provides a more consistent path loss exponent. Early experimental data confirms that fading rate is inversely correlated with directivity [20]. I thus do not expect fading to greatly impact the performance of my proposed protocol.

While the angle-of-arrival measurements can also be valuable in verifying a location claim, the angle-of-arrival measurements are susceptible to the reflection attack, in which an attacker uses a well-placed reflector to redirect

his transmission and makes himself appear to be located at the correct direction. I thus do not explicitly use the angle-of-arrival measurements in my verification protocol.

2.7.1 The Weak-Signal Attack

As stated in Section 2.2.1, the system noise of a verifier can be as high as -91.61 dBm. Thus, if a signal is weak when it reaches the verifier, it is possible that a benign prover is classified as a distance-enlarging attacker. Consequently, I propose requiring that the smaller of the measured signal strengths (and not the difference) must be much greater than twice the noise, or -88.60 dBm, in order for a verifier to accept or deny a response. However, from a security perspective, intentionally weakening one's signal is not beneficial because it does not improve the claim acceptance probability.

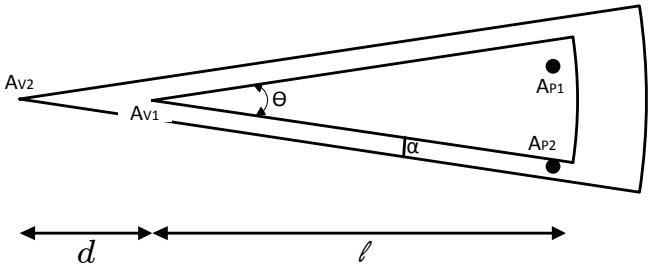
2.7.2 Attack Using Multiple Highly-Directive Antennas

The signal strength difference test relies on measuring the strength difference of a single signal. Thus, if a set of colluding attackers is able to transmit signals such that each of a verifier's antenna receives a different signal, then the colluders can deceive the verifier in a similar manner as deceiving the verification system.

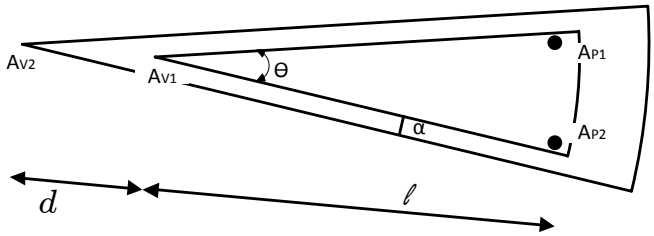
Figure 2.5(a) illustrates this attack. In this section, I analyze the plausibility of this attack. For simplicity, and only in this section, I assume the verifier uses sectored antennas with no side lobes. Let the beam-width of each sectored antenna be θ; then as shown in Figure 2.5(a), there exists, between the two antenna beam-cones, a gap of size $\alpha = d \sin \frac{\theta}{2}$.

Let an attacker be equipped with two antennas. The attacker's primary antenna A_{P1} responds to both the verifier's two directional antennas A_{V1} and A_{V2}. Then let the attacker's secondary directional antenna A_{P2} be able to compensate for the primary response at only the farther of the verifier's antennas A_{V2}. The attacker can then adjust the signal strength difference even if the attacker is closer than claimed.

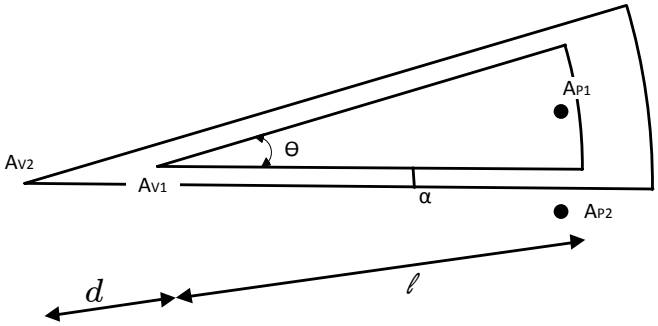
In other words, the attacker's secondary antenna A_{V2} must be smaller in dimension than α in order to fit within the gap of beam-cones at the verifier.



(a) Illustration of the feasibility of attack: α is large enough to fit an antenna



(b) Illustration of when the attacker does not know the exact verifier orientation and both attacker antennas are heard by both verifier antennas



(c) Illustration of when the attacker does not know the exact verifier orientation and the attacker's secondary antenna is outside of the beam-width of the verifier's secondary antenna

Figure 2.5: Illustration of the attack in which an attacker is equipped with two antennas and seeks to use the secondary antenna to compensate for the primary antenna

Suppose d is 0.562 m, which equals to 4.5 times the wavelength of a 2.4 GHz signal, and the verifier uses two directional antennas with beam-width of 10°, then the size of A_{V2} must be less than 0.05 m, or 0.4 times the signal wavelength. In order to radiate a signal, an antenna must be greater than one-tenth of the signal wavelength; thus this attack is indeed plausible.

However, the verifier can also patch this vulnerability by injecting randomness into the orientation of its directional antennas so long as the claimed location is within the beam-cones of both antennas A_{V1} and A_{V2}. This scheme is illustrated in Figure 2.5(b) and Figure 2.5(c). Consequently, the attacker cannot know the exact correct placement of his antennas, so his secondary antenna only affects one of two verifying antennas. Hence the verifier can detect and eliminate the attacker with high probability over multiple protocol rounds.

2.8 Evaluation

2.8.1 Methodology

I perform Monte Carlo simulations using MATLAB to study the effectiveness of simultaneous multilateration. In particular, I study the impact on claim acceptance from synchronization errors between verifiers. I also show that simultaneous distance bounding is more resilient to the collusion attack than naïve multilateration.

I simulate a distance bounding system with three verifiers, z apart from each other, forming a regular triangle. I let the uncertainty threshold δ be one-tenth of the distance between the verifiers and their geometric center ($\delta = 0.1\frac{z}{\sqrt{3}}$). From prior studies, I assume that each verifier suffers a time synchronization error that is normal-distributed with mean 0 ns. The synchronization error between each verifier's clock and the ground truth is within the larger of $3 \cdot 0.771 = 2.31$ ns and 4% of the distance between verifiers with 99.7% probability [17, 18]. Based on prior study [19], I simulate the case where each prover suffers from a uniformly-random processing delay between 0.8 and 1 ns.

To study the impact of synchronization on the multilateration performance, I simulate one prover that seeks to prove a location claim at the

geometric center of the verification triangle, i.e. $r = \frac{z}{\sqrt{3}}$, where the verifiers are 10 m $< z <$ 100 m away from each other. I let each prover be uniformly-randomly located x away from the claimed location, where $0.03 \leq \frac{x}{r} \leq 3$. For each simulation scenario, I perform 100,000 runs and calculate the average acceptance probability. The acceptance probability can be seen as a measure of the required attackers' effort to defeat a location verification system.

To study how resilient my proposed schemes are against the generic collusion attack, I consider the case in which the three verifiers are $z = 100$ m away from each other. I then let there be three colluding provers, uniformly-randomly located x away from the claimed location, where again $0.03 \leq \frac{x}{r} \leq 3$. I similarly calculate the acceptance probability when the verifiers use

1. naïve multilateration,
2. simultaneous multilateration, and
3. simultaneous multilateration with signal-strength difference measurement.

In my simulation, when the verifiers use signal-strength difference measurement to mitigate distance enlargement attack, each prover that is too close to verifier V_i uses a longer signal path of distance ℓ_i' to artificially enlarge the perceived distance:

$$\ell_i < \ell_i' = \frac{d}{\left(\exp\left(\frac{\gamma_{\min}}{\gamma_{max}} \ln\left(1 + \frac{d}{r}\right)\right) - 1\right)}.$$

The attacker's strategy is illustrated in Figure 2.6. In the illustration, the line-of-sight distance between the attacker and verifier V_3 is too short, and the attacker uses an alternate signal path ℓ_3' (for example, by setting up a curved waveguide) that satisfies the above inequality. In my simulation, I consider a scenario where the path loss exponent is relatively consistent, and let $\gamma_{\max} = 2.5$ and $\gamma_{min} = 2$.

To evaluate collusion resilience, I consider not only the acceptance probability, but also the normalized distance between provers and the claimed location (x/r) beyond which all location claims are rejected. That is, $\xi = \arg\min\left\{\frac{x}{r} \mid \text{all location claims are rejected}\right\}$. ξ is an important factor in determining how far a set of sophisticated colluders can reside and still deceive the verifiers.

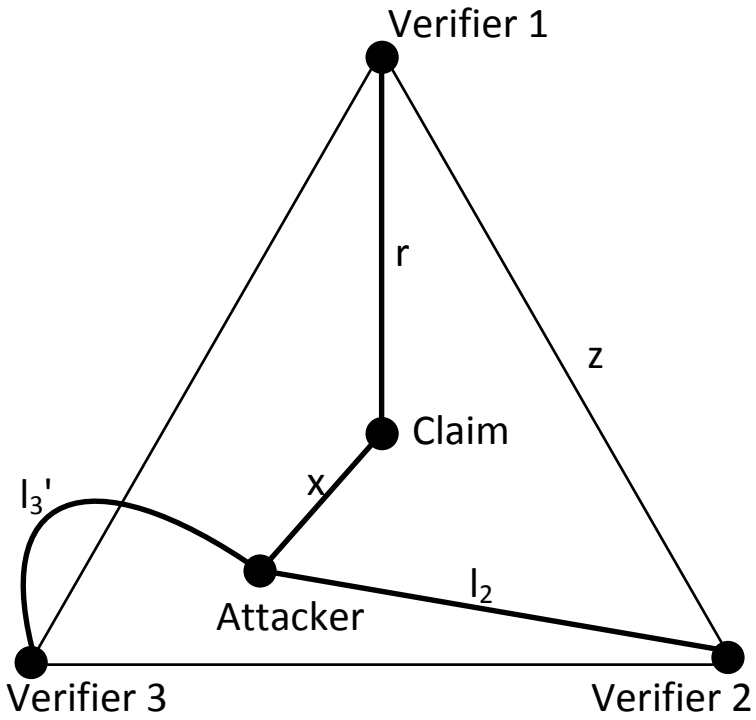


Figure 2.6: Illustration of the attack strategy

2.8.2 Simulation Results

When there is a single prover, my simulation result is shown in Figure 2.7. For naïve multilateration, I only show the results when verifiers are 10 m apart. This is because

1. Each verifier performs distance bounding separately; hence verifier-specific synchronization errors do not impact the outcome; and

2. The uncertainty threshold is chosen to be the distance between verifiers multiplied by a constant.

Thus, if verifiers are farther than 10 m apart, the provers' processing delays contribute less to the measured uncertainty, and the acceptance probability increases.

To show the impact of verifier synchronization errors, I draw a different line for each choice of the distance between verifiers. I show the acceptance probability on the y-axis, and the normalized prover deviation x/r on the x-axis. I observe that the performance of simultaneous multilateration, taking

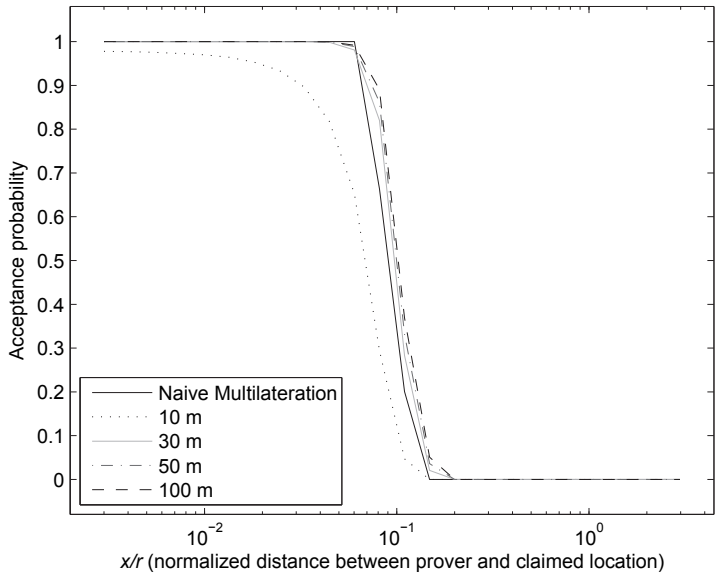


Figure 2.7: Probability that a location claim is accepted given the prover is x away from the claimed location

into account the synchronization errors between verifiers, is comparable to the naïve multilateration.

When there are three colluding attackers, I show my simulation result in Figure 2.8. To study the performance difference, I draw a different line for each verification protocol. I observe that when three colluding attackers are present, naïve multilateration might accept the location claim even when the attackers are *outside* the convex hull of the verifiers ($\xi > 2$). This shows that collusion can significantly impact the security of naïve multilateration. Simultaneous multilateration mitigates the collusion attack so that the colluding attackers must be significantly more sophisticated to deceive the verifiers. However, the colluders can still successfully attack while outside the convex hull formed by verifiers ($\xi \approx 1.3$). Finally, by using signal-strength-difference to mitigate the distance enlargement attack, the verifiers can reject all claims made by provers located $x > 0.5r$ ($\xi \approx 0.5$) away from the claimed location. In other words, in this example topology, signal-strength-difference-test offers the strong property that the colluders cannot successfully attack once outside the convex hull formed by the verifiers.

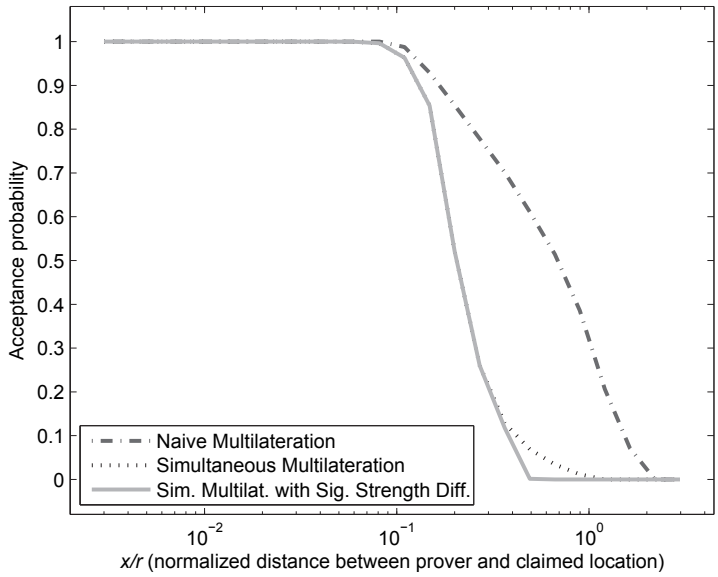


Figure 2.8: Probability that a location claim is accepted given three colluding attackers are all x away from the claimed location

2.9 Chapter Summary

The distance bounding protocol provides a strong result in verifying that a prover is within a certain distance from a verifier. However, in order to provide reliable position-based service, the network must securely verify the user's precise location information. In this chapter, I propose a verification protocol that is based on simultaneous multilateration. I show that, without compromsing the false alarm rate, the proposed protocol achieves the highest detection rate of false alarm claims achievable by any verification system based solely on time-of-flight measurements.

I also show an extension of the simultaneous multilateration protocol that can detect and eliminate the distance enlargement attack. By eliminating the distance enlargement attack, the simultaneous multilateration protocol can completely mitigate the generic collusion attack.

CHAPTER 3

PARTIAL DEAFNESS: A MAC LAYER ATTACK THAT EXPLOITS DIFFERENT NOTIONS OF FAIRNESS

Wireless networks based on the IEEE 802.11 standard [21] are widely deployed today for governmental, commercial, and personal uses. Attacks against the 802.11 standard can cause widespread security issues ranging from mere inconvenience to privacy breaches and machine compromise. Much attention is dedicated to both possible attacks and their respective solutions.

Heusse et al. demonstrate that even without any malicious intent or misbehavior, a slow connection can still significantly impact the transfer speed of a fast connection because of the fairness mechanism implemented by the Distributed Coordination Function (DCF) at the Medium Access Control layer (MAC) [22]. In particular, since the IEEE 802.11 DCF seeks to fairly grant access opportunities to each station, each station has an equal opportunity to be the next station to transmit a data packet; thus a fast connection regularly has to wait until a slow connection finishes its reception. This performance anomaly together with excessive channel reservation can be viewed as head-of-queue blocking at the wireless medium since the DCF cannot schedule the next station until the current transmitter is finished.

In this chapter, I present the *partial deafness attack*, a DoS attack that builds on the observation of Heusse et al. [22]. Partial deafness is based on the fact that fair access opportunities may result in significantly unfair channel access durations. My attack is based on the realization that most commercial access points are implemented with only a single data queue because the 802.11 standard does not specify or recommend any queuing behavior. Thus, if a transmitted packet is not acknowledged, the packet triggers retransmissions and possible rate adaptation (i.e. slowing the data rate), thereby creating head-of-queue blocking at the access point. The head-of-queue blocking then drastically degrades the performance of the wireless network.

Like other DoS attacks, the partial deafness attack does not aim to give better performance to the attacker, but to reduce the performance of other users. In the partial deafness attack, each attacker *artificially worsens* his link quality by intentionally failing to acknowledge packet receptions. The partial deafness attack impacts the system in a manner similar to a legitimate user with a slow connection. However, by exploiting the retransmission mechanism specified by the 802.11 standard, the impact of our attack becomes much more devastating, especially to the Transport Control Protocol (TCP) performance of other users.

The partial deafness attack targets the MAC-layer protocol but does not require the attacker to modify the MAC protocol implementation at his station. For example, an attacker can suppress an acknowledgment by turning off the network interface card any time between the start and completion of packet reception. In Section 3.4.1, we detail our implementation of a partial deafness attacker by enabling and disabling the acknowledgment function in the driver of a commercial Wireless Local Area Network (WLAN) card. In other words, our attack works even when the attacker abides by the same MAC rules as every other node.

An attacker can simply move farther away from the access point to physically worsen his channel condition and impact other users. However, this approach requires the attacker to find a location such that the channel condition is sufficiently weak to regularly result in retransmission, and yet is not weak enough to result in disassociation. If fading causes the attacker to be disconnected, then the attacker cannot impact other users; on the other hand, if fading improves the attacker's channel condition intermittently, then other users can also intermittently experience improved transfer rate.

Since the partial deafness attack exploits the fairness mechanism and the head-of-queue blocking at the access point, many different methods can mitigate the attack. In this chapter, I propose enforcing time-fairness at the access point instead of relying on a single First-In-First-Out (FIFO) data queue. My proposed scheme can be implemented entirely in software, and does not require any changes to the widely used 802.11 MAC protocol. The result shows that different notions of fairness can result in significantly different network performance; thus, when designing a fairness-enforcing DoS mitigation protocol, the designer must be careful with the choice of fairness metric.

3.1 Overview of IEEE 802.11

The IEEE 802.11 standard [21] in its various forms is now widely implemented and deployed in wireless networks. In *infrastructure mode*, users share the wireless medium in order to download and upload data via an *access point*. The IEEE 802.11 standard specifies multiple transmission rates, which a station can choose based on the signal strength and link quality between the communicating stations. There is a natural trade-off between the transmission rate and error rate dictated by the rate distortion theorem [23]. Rate adaptation such as Auto Rate Fallback (ARF) [24] and SampleRate [25] have been widely implemented to adaptively choose an optimal transmission rate as link quality varies. Rate diversification is thus common in any 802.11 network.

At the Medium Access Control (MAC) layer, the IEEE 802.11 standard specifies methods to *fairly* share the wireless link among users. The Distributed Coordination Function (DCF) is the predominant MAC protocol in today's wireless networks. It seeks to fairly share the medium by offering equal access opportunity to all users. Another MAC protocol is specified in the 802.11 standard: the Point Coordination Function (PCF); however, few vendors implement the PCF.

The DCF provides access opportunity fairness by using *random backoff*. Each station in 802.11 maintains a *Contention Window* (CW). After any broadcast transmission or successful unicast transmission, the contention window is reset to $\mathrm{CW}_{\min}$, and it never increases beyond a value $\mathrm{CW}_{\max}$. When station A wishes to send a frame, it first performs carrier sensing for the *DCF InterFrame Space* (DIFS)[1] to determine whether or not the medium is busy. If the medium is not busy for that entire period, then A can immediately send the data frame; otherwise A picks a random backoff value uniformly distributed over interval $[0, \mathrm{CW}]$, where CW is the current value of the contention window. Each time the medium is no longer busy, A waits DIFS, and if the medium is still not busy, decrements its backoff value by one for every slot time the medium continues to be free. When A detects that the channel is once again busy, it stops decrementing the backoff value and repeats the process of waiting for the channel to become idle, waiting

[1] DIFS = SIFS + 2 × Slot Time; SIFS and Slot Time are listed in Table 3.1

Table 3.1: Slot time, Short InterFrame Space (SIFS), minimum and maximum contention window (CW) values of the corresponding transmission mode in the physical layer: frequency hopping spread spectrum (FHSS), direct sequence spread spectrum (DSSS), infrared (IR), and orthogonal frequency-division multiplexing (OFDM)

PHY mode	Slot Time	SIFS	$\mathrm{CW}_{\mathrm{min}}$	$\mathrm{CW}_{\mathrm{max}}$
FHSS	50 μs	28 μs	15	1023
DSSS	20 μs	10 μs	31	1023
IR	8 μs	10 μs	63	1023
OFDM	9 μs	16 μs	15	1023

DIFS, and counting down its remaining backoff value. Once A's backoff value reaches zero, A starts its transmission.

When station B successfully receives a unicast transmission from A, B waits for a *Short InterFrame Space* (SIFS), and, if the channel is idle, immediately returns an acknowledgment. Thus, if a station A sends a unicast transmission and does not receive a MAC-layer acknowledgment for that transmission, A will believe that the packet did not successfully reach B (broadcast messages are not acknowledged so they are always considered successful; previous work has shown that such packets are therefore delivered less reliably [26]). A will then retransmit the unsuccessful frame following an exponential backoff procedure. The contention window for the n^{th} transmission is given by

$$\begin{aligned} \mathrm{CW}[1] &= \mathrm{CW}_{\mathrm{min}} \\ \mathrm{CW}[n] &= \min\left(2\mathrm{CW}[n-1]+1, \mathrm{CW}_{\mathrm{max}}\right) \end{aligned}$$

and A's *station retry count* is incremented by one. If the station retry count exceeds the retry limit,[2] the frame is discarded; otherwise A chooses a new backoff value on the interval $[0, \mathrm{CW}[n]]$ and repeats the process of waiting for DIFS and counting down the backoff time as it did for the original transmission.

The slot time, SIFS, and contention window are dictated by the transmission mode at the physical layer. Table 3.1 shows various physical layer

[2]The 802.11 standard specifies two retry limits, ShortRetryLimit = 7, and LongRetryLimit = 4; typical packets are retransmitted with ShortRetryLimit

transmission modes with their corresponding slot time, SIFS, and contention window.

3.2 Related Work

The IEEE 802.11 standard is widely deployed due to the unlicensed spectrum in which it operates and the low cost of client devices and access points. As a result, the security of 802.11 attracts much attention. In particular, most research on MAC security focuses on the requirements of confidentiality and integrity. The original security protocol, Wired Equivalent Privacy (WEP), is designed to provide privacy and authenticity of data. However, Fluhrer et al. note that weakness in the encryption algorithm used by WEP can be exploited to allow the discovery of session keys [27]. Numerous related attacks are described in the literature [28, 29].

While a cryptographic attack has strong adverse effects on users' privacy and a protocol's confidentiality and integrity, our work considers another type of attack where the attacker seeks only to deny service to other users. That is, the attacker aims to reduce a protocol's availability. Specifically, we consider the attacks against the MAC-layer protocol specified in 802.11 rather than the pure resource consumption attacks such as the jamming attack.

Attacks on the 802.11 MAC protocol can exploit management vulnerabilities. Bellardo and Savage implement and demonstrate an attack that targets the authentication/association scheme of 802.11 [30]. Bellardo and Savage note that the deauthentication and disassociation messages are not encrypted; thus an attacker can easily forge these messages. The attacker can then send the deauthentication message to the access point before the client's data is received, or the attacker can send the disassociation message to the client before the client's data is transmitted. Ferreri et al. [31] describe DoS attacks against an access point's association and authentication mechanisms.

Attacks on the 802.11 MAC can also exploit media access vulnerabilities. Bellardo and Savage also note that the 802.11 carrier sense mechanism can be easily exploited. For example, in 802.11 networks, a node can only send data during a certain time period after the channel stops being busy. In particular, if not due to retransmission or fragmentation, a user can only

transmit data DCF InterFrame Space (DIFS) after the channel is available; otherwise the user can transmit data Short InterFrame Space (SIFS) after, where SIFS $<$ DIFS. A very simple method to deny service is to send a short burst every SIFS. Bellardo and Savage present a more sophisticated scheme exploiting the virtual carrier sense mechanism. The 802.11 standard specifies that the MAC frame header of all packets should contain a *duration* field, which specifies how long others have to wait before transmission is allowed in order to avoid collision. Users update their Network Allocation Vector (NAV) with this duration information and keep quiet for the specified duration. Thus an attacker can repeatedly request long channel occupancy time, thereby starving normal clients of channel occupancy.

The benefit of attacking the duration field rather than sending a short burst every SIFS is the low amount of power used to carry out the attack. In the duration field attack, an attacker simply initiates a Request to Send (RTS)/ Clear to Send (CTS) handshake along with the specified duration. The handshake in theory would keep the channel busy for roughly 30 ms. The short burst approach, on the other hand, requires sending a short burst every SIFS, or 10 μs in 802.11b/g networks. My proposed attack performs even better in terms of power saving for the attackers; in particular, the partial deafness attack can easily occupy 100 ms of channel time without having to send any messages. Moreover, the partial deafness attack does not require the attacker to have better service, higher power, or closer distance to the access point. Finally, the partial deafness attack works on every access point tested; however, the duration field attack does not work in many real systems because most vendors do not implement the 802.11 specification correctly [30].

Heusse et al. point out that when a client uses a lower bit rate than others in an 802.11 network, the performance of all clients is considerably degraded [22]. Tan and Guttag subsequently suggest that time fairness can mitigate this performance anomaly and provide better throughput for the WLAN [32]. In this chapter, I present an attacker that exploits the conclusion of Heusse et al. by artificially and intentionally creating rate disparities. I show that access point retransmissions exacerbate the anomaly by creating head-of-queue blocking at the access point's data queue. I then adapt the principle of Tan and Guttag's solution and show how to mitigate the partial

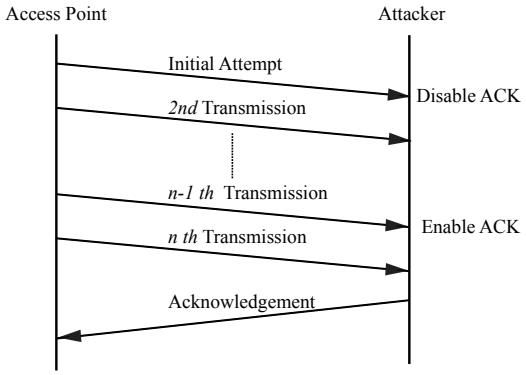


Figure 3.1: Partial deafness attack

deafness attack by implementing time fairness at the access point's data queue.

3.3 The Partial Deafness Attack

3.3.1 Description

In this section, I present the *partial deafness attack*, which partly exploits the retransmission mechanism of the 802.11 protocol to reduce the bandwidth of non-attacking nodes. In the partial deafness attack, the attacker, upon receiving a unicast data frame addressed to it, intentionally fails to send a timely acknowledgment for at least a portion of those data frames. Though previous work has suggested denial-of-service attacks against IEEE 802.11 by exploiting management vulnerabilities, my proposed attack stands out because it substantially reduces the bandwidth available to legitimate nodes without requiring the attacker to have superior connection quality. That is, an attacker with lower transmission power, fewer computation resources, and located farther away than a normal client can still significantly degrade service to all the normal clients within the network.

As illustrated in Figure 3.1, when a unicast transmission is not acknowledged, an 802.11 station will normally transmit a frame up to seven times before it gives up and discards the frame. An attacker can thus fail to acknowledge the first six transmissions. In addition, senders in 802.11 employ *rate adaptation* (e.g. Auto Rate Fallback (ARF) [24], or SampleRate [25]) to maximize the throughput of the channel. When a receiver repeatedly fails

to receive transmissions at one bit rate, the sender chooses a lower bit rate in an attempt to successfully deliver the packet. Eventually the sender will choose the lowest possible rate, called the *base rate*, to deliver packets to the attacker.

Since most 802.11 networks are infrastructure networks in which clients connect directly to an access point, and most traffic is directed to or received from an access point, the behavior of an access point plays an important role in the fairness perceived by a station. The 802.11 standard does not specify or recommend any queuing behavior at the access point, so most commercial access points use a single queue. Thus all packets are treated with the same priority and each packet is completed before subsequent packets can be serviced, regardless of the number of retransmissions or the rate that is selected for those retransmissions. The attacker can thus induce the access point to spend a large amount of time to transmit to the attacker, thereby drastically decreasing the time allocated to the normal clients and reducing the overall throughput.

3.3.2 Analysis

I first analyze the impact of the partial deafness attack in 802.11b, where the maximum rate is 11 Mbps and the base rate is 1 Mbps. I then use a theoretical analysis to show that rate diversity exacerbates the problem; thus, commonly deployed 802.11b/g networks, in which the maximum and base rates are 54 Mbps and 1 Mbps, respectively, are even more susceptible to our attack.

To quantify the degree of imbalance caused by the partial deafness attack, we consider a case in which a normal client and a malicious client share one base station. We call the normal client Alice; the malicious client, Mallory; and base station, Bob. In our example, Alice and Mallory have the same link quality to Bob, so when Mallory is not performing any attack, Bob can send to both Alice and Mallory at 11 Mbps. That is, if Alice and Mallory started User Datagram Protocol (UDP) downloads, they would each receive approximately half of the available bandwidth.

Consider the particular rate adaptation algorithm implemented on a Linksys WRT54G access point. Initially, Bob's rate adaptation chooses 11 Mbps for

its first three transmissions and 2 Mbps for its last four retransmissions. If Mallory acknowledges after the third transmission, Bob determines that 11 Mbps is too high an initial rate, and will send the subsequent packet at 5.5 Mbps for the first three transmissions and 1 Mbps for the next four retransmissions. If Mallory again acknowledges after the third transmission, Bob determines that 5.5 Mbps is again too high an initial rate, and will send the subsequent packet at 2 Mbps for the first three transmissions and 1 Mbps for the next four retransmissions. If Mallory again acknowledges after the third transmission, Bob will determine that 2 Mbps is still too high and will send all subsequent packets at 1 Mbps.

If Mallory performs the partial deafness attack, and she does not acknowledge receiving a packet until the seventh transmission, Bob would send packets to Mallory at 1 Mbps in the steady state, but to Alice at 11 Mbps. Thus, it would take Bob 11 times longer to send an identical packet to Mallory than to Alice. In other words, if Bob sends an equal number of packets to Alice and Mallory, without considering retransmission, Mallory is already allocated $\frac{11}{12} = 91.7\%$ of the channel occupancy time as opposed to 50% in a time-fair scheme.

Now consider the additional effect of retransmissions. In the direct sequence spread spectrum (DSSS) mode of 802.11b, the slot time is 20μs, and the minimum and maximum contention window (CW) sizes are 31 and 1023, respectively. Typically 802.11 networks are configured to allow a maximum transmission unit of around 2304 bytes. In 802.11, a station can fragment larger packets into smaller fragments and transmit each fragment separately. In this case, Mallory allows Bob to send each fragment the maximum number of times before Bob gives up on the fragment. Thus each fragment of the packet is transmitted seven times, which is nearly equivalent to transmitting the entire packet seven times. (There are minor differences because of the interframe spacing used between fragments; but seven retransmissions of one large frame should closely approximate seven retransmissions of each of several smaller fragments.)

Now let us quantify Mallory's per-packet channel occupancy time in steady-state. Let us assume that every time the sender (in this case Bob) wishes to send a packet, the medium is busy, so the first transmission experiences backoff. Further assume that once the medium becomes idle, there are no further transmissions on that medium except those initiated by Bob. I vali-

date the theoretical results here with implementation results in Section 3.4.1, which show that these assumptions provide results comparable to those seen in normal access point behaviors. I consider a single UDP packet containing 1470 bytes of data, which, after UDP- and Internet Protocol (IP)-layer headers, comes to 1498 bytes. The addition of MAC-layer headers brings the total to 1534 bytes.

If Alice and Mallory both acknowledge reception of a packet by the third transmission, the steady-state data rate is 11 Mbps. In this case, the first transmission takes about 1571.6 μs in expectation: 50 μs for DIFS, 310 μs of expected backoff, 96 μs of preamble, and 1115.6 μs of data. Bob would expect an acknowledgment within 126 μs, which represents the sum of the SIFS that Mallory must wait following reception, the maximum propagation delay between Mallory and Bob, which is defined in 802.11 to be one slot time, and the delay that 802.11 allows between when the radio frequency energy starts impinging on the receiver until that receiver starts receiving a message, which is defined to be the length of the preamble. In expectation, a failed first transmission would therefore be detected 1697.6 μs after the medium becomes idle. When the first transmission is successful, Mallory waits SIFS and transmits a preamble and a 12 byte acknowledgment at 2 Mbps, which gives an expected time of 1725.6 μs from when the medium is idle until the transmission is received. (We assume the propagation time is negligible; the 20 μs slot time of 802.11 is sufficient for a 6 km transmission, which is well in excess of typical 802.11 transmission distances.) In further retransmissions, the one thing that changes is the expected backoff value, which increases from 310 μs to 630 μs to 1270 μs within these first three retransmissions. Also, Bob will not wait DIFS when Bob does not receive an acknowledgment. Thus success after three retransmissions takes $1697.6 + (1647.6 + 320) + (1675.6 + 320 + 640) = 6300.8$ μs. If Mallory forces three retransmissions for each packet while Alice acknowledges every first transmission, then Mallory will capture $\frac{6300.8}{6300.8+1725.6} = 78.5\%$ of the channel occupancy time.

When Bob must regularly transmit each packet at least four times in order to reach Mallory, Bob sends every packet to Mallory at 1 Mbps. Thus each data transmission takes 12272 μs for data alone, which, after adding backoff, preamble, and header for the first transmission, takes 12678 μs. The acknowledgment times out after the same 126 μs, giving a failure time for the first transmission of 12804 μs. Thereafter, each failure takes the same

amount of time after adjustment for backoff, and when the acknowledgment finally comes, it is transmitted at 1 Mbps, so seven retransmissions takes $50 + 12678 * 7 + 126 * 6 +$ backoff increases $+ 202$ (μs), where 50 μs is DIFS, 12678 μs is the time that each packet transmission takes, 126 μs is the time to detect that an acknowledgment is not forthcoming, and 202 μs is the time to finish receiving an acknowledgment. The total additional backoff for seven retransmissions is 28160 μs in expectation, so the total transmission time is 117914 μs. If Mallory forces six retransmissions (for a total of seven transmissions) for each packet while Alice acknowledges every first transmission, then Mallory will capture $\frac{117914}{117914+1725.6} = 98.6\%$ of the channel occupancy time.

Finally, the rate diversification exacerbates the partial deafness attack. In the same scenario, when Alice uses a 54 Mbps link in a 802.11b/g network, Mallory's transmissions take the same amount of time, but Alice's transmissions are now much faster. The DIFS and backoff take 360 μs as before (because it is a mixed-mode 802.11b/g access point), 802.11g does not require a preamble, and Alice's data transmission is now 227.3 μs, for a forward transmission time of 587.3 μs; after a 10 μs 802.11g SIFS and a 30 μs 802.11g acknowledgment, each of Alice's packets take 627.3 μs in expectation. Thus Alice's channel occupancy time drops further to 0.53%.

3.4 Implementation and Evaluation of the Attack

3.4.1 Implementation

In this section, I detail one implementation of a partial deafness attacker and observe the effect of the attack on an 802.11 network. My implementation uses commercial off-the-shelf 802.11 Network Interface Cards (NICs). Most commodity 802.11 NICs generate and send acknowledgment frames automatically in firmware whenever a packet is received, because of the hard real-time deadlines on generating acknowledgments. The partial deafness attack can then be implemented by building custom hardware, modifying the firmware to defer acknowledgments, or turning off the network interface card any time between the start and completion of packet reception.

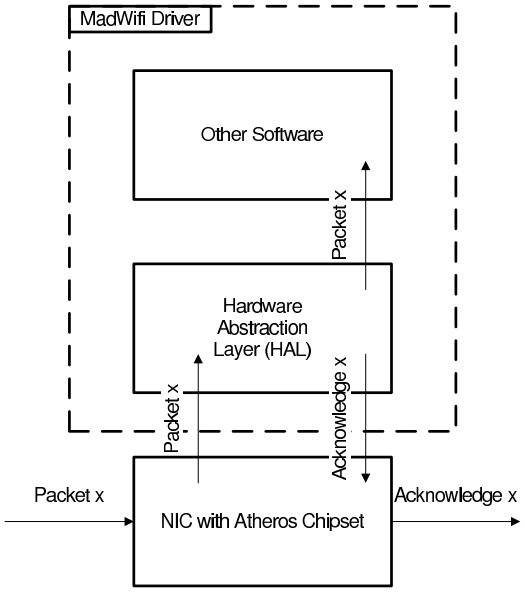
In order to simplify the task of deferring packet acknowledgments, I choose to modify the MadWifi driver, which is a Linux kernel device driver for Atheros-based WLAN devices. The Atheros chipset does not load a firmware onto the card, but instead relies on a Hardware Abstraction Layer (HAL) module that is part of the driver. The HAL module defines the interface between the hardware and other software in the device driver to manage many of the chip-specific operations and to enforce any relevant regulations. The normal operation of the Atheros card is illustrated in Figure 3.2(a).

My implementation modifies MadWifi to control a particular register in the HAL module, which in turn, enables and disables packet acknowledgments. The modified behavior of the Atheros card is illustrated in Figure 3.2(b). As illustrated in Figure 3.1, we suppressed acknowledgements from the first $n - 1^{\text{th}}$ transmissions by switching the HAL register.

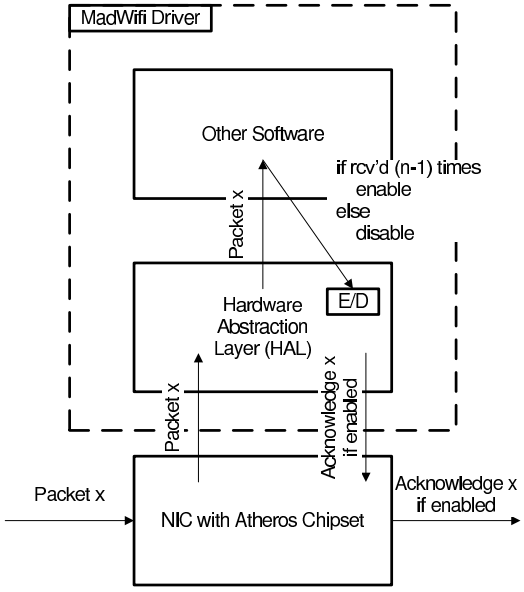
To evaluate the effectiveness of a partial deafness attacker, I consider an evaluation network consisting of a traffic source connected to an IEEE 802.11b/g access point. A normal user and an attacker use 802.11 to connect to the access point. This topology is illustrated in Figure 3.3. To evaluate the effectiveness of the attacker, I use commercial off-the-shelf access points such as Linksys WRT54G, which uses the Broadcom BCM5352EKPB chipset and supports 802.11b/g mixed mode, because it shows how the rate adaption is practically implemented in a real 802.11 system. The experiment uses MadWifi and configures the Atheros NIC to operate in 802.11 master mode. The experiment then uses kernel-level bridging to bridge between the 802.11 network interface card and the Ethernet network interface card. A traffic source generates traffic as an iperf client, which was then sunk at iperf servers running on the normal user and the attacker. An additional machine (not shown in Figure 3.3) collects data by capturing all 802.11 frames sent on the network.

3.4.2 Evaluation of the Partial Deafness Attack

Maximum Throughput of Attacker In order to determine an attacker's minimum bit rate that can saturate the channel, I first examine the maximum throughput of the attacker using 802.11b when the attacker is the only user of the access point. These measurements and theoretical analysis use UDP



(a) Normal operation of an Atheros NIC



(b) Modified operation of an Atheros NIC

Figure 3.2: Illustration of Atheros NIC operation. Events happen from left to right.

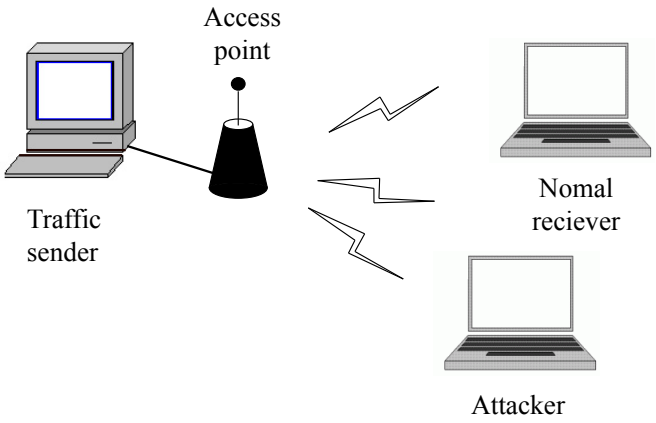


Figure 3.3: Network topology

Table 3.2: Maximum UDP throughput of an attacker. n is the number of transmissions required before the attacker sends an acknowledgment; this table shows results in a theoretical analysis as described in Section 3.3 and an actual outdoor/indoor experiment without/with any detectable 802.11 interference.

n	Theoretical	Outdoor	Indoor
1	6814.9 (kbps)	6049.0 (kbps)	5782.0 (kbps)
2	3184.2 (kbps)	N/A	N/A
3	1866.4 (kbps)	1563.0 (kbps)	1282.0 (kbps)
4	214.4 (kbps)	209.1 (kbps)	193.3 (kbps)
5	162.3 (kbps)	163.2 (kbps)	159.4 (kbps)
6	123.5 (kbps)	128.8 (kbps)	123.2 (kbps)
7	99.7 (kbps)	115.0 (kbps)	114.0 (kbps)

flow because UDP is a non-conforming load and will allow an attacker to set the load regardless of the route's capability to handle that load. When Mallory forces Bob to transmit each packet n times, I compute the amount of time required per packet as described in Section 3.3; I then translate it into an application-layer rate and present it in Table 3.2.

As described in Section 3.3, the rate adaptation mechanism at the access point selects an 11 Mbps rate for users that acknowledge at least once every three transmissions and selects a 1 Mbps rate for users that acknowledge less frequently than every three transmissions. This contributes to the sharp reduction in maximum throughput between a user who acknowledges every three packets and a user who acknowledges every four packets.

I then implement a partial deafness attacker that requires $n \leq 7$ transmissions before it sends an acknowledgment. The driver that enables and

Table 3.3: UDP throughputs under partial deafness attack. Attacker's source rate is 200 kbps. Results are averaged over 20 runs.

Normal user's source rate	Normal user's throughput	Attacker's throughput
100 (kbps)	55.7 (kbps)	112.0 (kbps)
200 (kbps)	111.8 (kbps)	111.7 (kbps)
400 (kbps)	219.1 (kbps)	109.3 (kbps)

disables acknowledgments could not consistently set the register within the real-time requirement between the first and the second transmissions; thus my experiment does not test the case when $n = 2$. The experiment ran this attacker both in an outdoor environment without measurable 802.11 interference and in an indoor environment where the 802.11 interference was uncontrolled. Some experimental results are greater than the calculated theoretical values because the access point, in violation of the specification, interleaves a beacon transmission between retransmissions of the original data packet. Because beacons are broadcast, and because broadcast messages are always considered successful, the access point resets the contention window size to minimum without resetting the retry count. My results show that a partial deafness attacker receiving about 115 kbps of traffic can exhaust the entire forwarding capability of an access point.

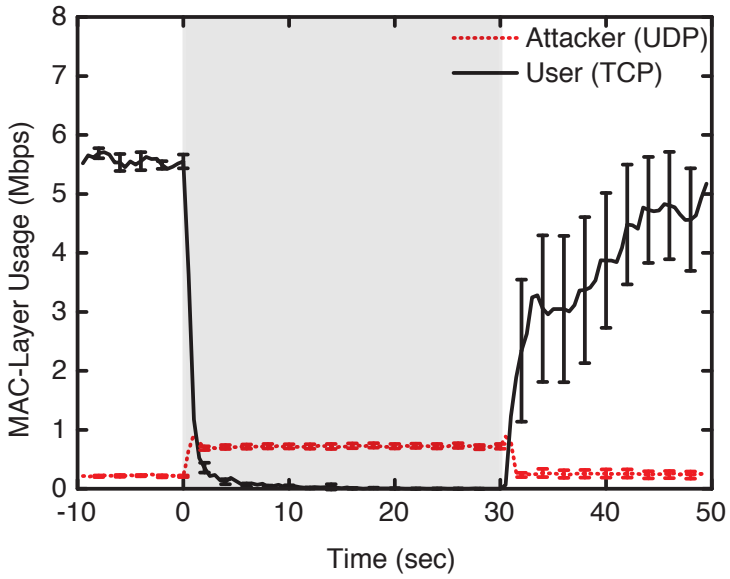
Impact on UDP Victim I then consider the impact on the throughput of a normal client that uses UDP against a partial deafness attacker that only acknowledges the seventh transmission of each packet. Theoretically, if the access point receives α packets destined to the normal user for every packet destined to the attacker, then we expect that the normal user would get a $\frac{\alpha}{1+\alpha}$ share of the overall throughput, since the access point treats all packets equally.

To test this hypothesis, the attacker downloads at a UDP source rate of 200 kbps, which is sufficient to saturate the access point's wireless link under the partial deafness attack; and the normal user downloads at a UDP source rate of 100, 200, then 400 kbps. The resulting throughput is shown in Table 3.3. As expected, the ratio of throughputs is equal to the ratio of the UDP source rates.

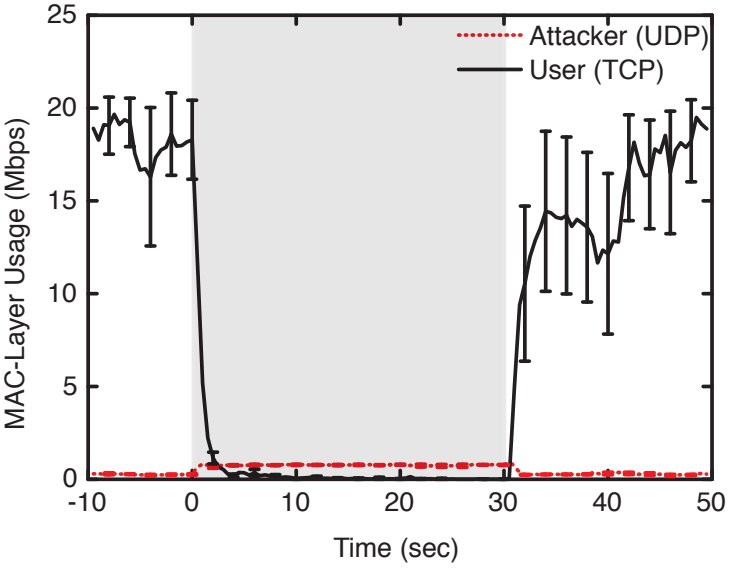
Impact on TCP Victim I then consider a normal TCP user competing for bandwidth against a partial deafness attacker. The attacker again downloads at a UDP source rate of 200 kbps. To show the impact of the attack, I allow the TCP flow to warm up for a period of time before the attack starts; I then perform the attack for a period of time, and finally turn off the attack and allow TCP to return to its steady-state behavior. I measure the MAC-layer bandwidth usage to study how nodes share the available bandwidth on the wireless link. The MAC-layer measurement counts each retransmission as additional channel use. As shown previously, each transmission to the attacker theoretically takes around 118 ms. I thus quantize each protocol's usage into 500 ms slots so that the normal user has a chance to receive data in each slot, and each slot conveys the granularity of MAC-layer usage. I plot the MAC-layer usage over time for each scenario. Because the TCP flow has a warm-up and cool-down period where the attacker does not perform the partial deafness attack, each plot includes a shaded box covering the 30-second time interval (from 0 to 30) during which the attack took place.

Figure 3.4(a) shows the MAC-layer usage when a partial deafness attacker competes against a normal user's TCP flow when both clients use 802.11b. As shown in Table 3.2, a UDP attacker only needs to transmit 115 kbps in order to saturate the link and cause congestion; allowing the attacker to send 200 kbps traffic would cause the attacker to experience a 43% loss rate without considering a sharing normal user. When a normal TCP user shares the channel with the attacker, the access point treats and drops an equal fraction of UDP and TCP packets; hence the TCP user would experience a similar loss rate as the attacker. That is, the normal TCP user would experience at least a 43% loss rate; since TCP is a conforming transport layer protocol, such a high loss rate causes repeated TCP time-out and results in minuscule throughput for the normal user, as shown in Figure 3.4(a).

I then examine the impact of a partial deafness attacker in the scenario where a normal user connects to the access point using the 802.11g standard. The normal user enjoys a faster connection when the attacker is silent; however, when the attacker carries out the partial deafness attack, the transfer speed of the normal 802.11g user is not significantly faster than that of a normal 802.11b user, as illustrated in Figure 3.4(b). This result is consistent with our analysis of rate diversity in a 802.11b/g network at the end of Section 3.3.



(a) Impact on 802.11b normal user



(b) Impact on 802.11g normal user

Figure 3.4: MAC-layer utilization by TCP under the partial deafness attack. The shaded region (0-30 sec) shows the time of attack; results are averaged over 20 runs, with the error bars showing 95% confidence interval.

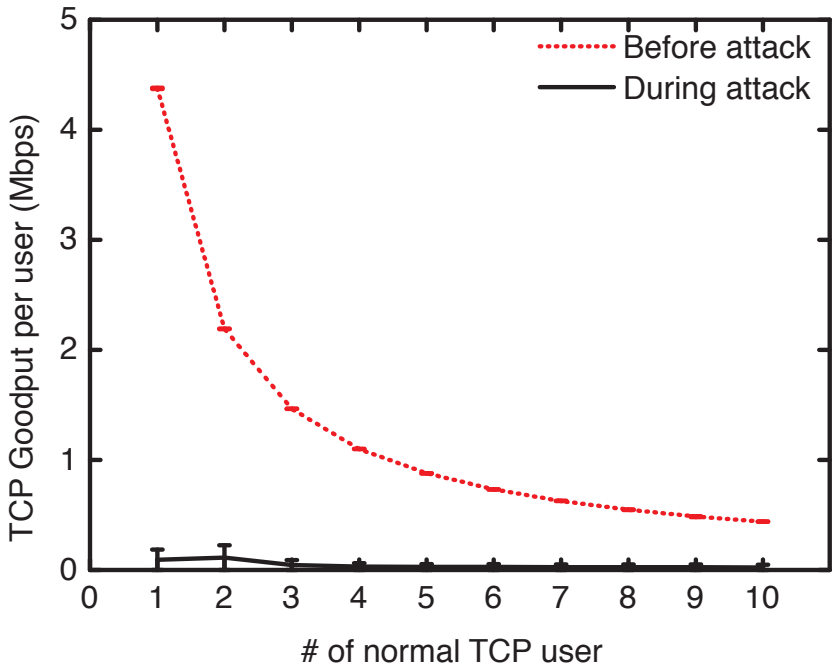


Figure 3.5: Ns-2 simulation of the partial deafness attack on a network with multiple 802.11b normal users and an UDP attacker. Results are averaged over 20 runs, with the error bars showing 95% confidence interval.

The partial deafness attack creates head-of-queue blocking by using retransmission and rate diversity; thus, a normal user will experience an even higher loss rate when other normal users are also present. This is intuitive since all users are going to compete for the limited amount of remaining bandwidth.

I also perform an ns-2 simulation on the impact of the partial deafness attack in a network with 1 to 10 normal users in addition to the attacker. In the simulation, all users (normal and attacker) are located on a circle 1 m away from the access point. The normal users and the attacker are given identical properties (such as signal and noise power levels), except the acknowledgment policy. That is, the attacker is identical to a normal user except he does not acknowledge receiving a packet until the seventh transmission. Figure 3.5 shows the effectiveness of the partial deafness attack when the attacker uses UDP with source rate of 200 kbps. The goodput per normal user during attack is minuscule compared to the fair goodput each normal user enjoys without the attack.

I examine the effectiveness of the partial deafness attack on two other access points that use different chipsets from that of Linksys WRT54G. Specifi-

cally, I examine a Linksys WRT54GC, and a Trendnet TEW-432BRP access points. Figure 3.6 shows the result, and both access points are also susceptible to the partial deafness attack. Even though rate adaptation mechanisms of these two access points are different from that of Linksys WRT54G, the partial deafness attack still makes the attacker's traffic use the base rate during attack period. For the Linksys WRT54GC, each packet is retransmitted only four times. The rate adaptation mechanism in Trendnet TEW-432BRP decreases the rate slowly as compared to the Linksys WRT54G. This difference results in slower performance degradation, as shown in Figure 3.6(b).

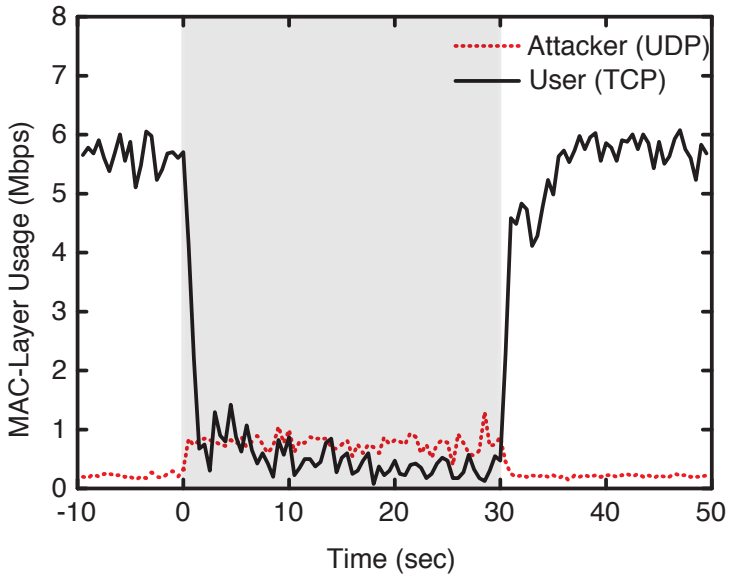
3.5 Countermeasure

In this section, I propose a countermeasure that mitigates the partial deafness attack. The partial deafness attack is based on head-of-queue blocking at the access point that results in starvation of normal users. Thus I propose mitigating the attack by instead enforcing time fairness to prevent starvation. Time fairness has also been suggested in previous work [32] to increase throughput in a network with rate diversity.

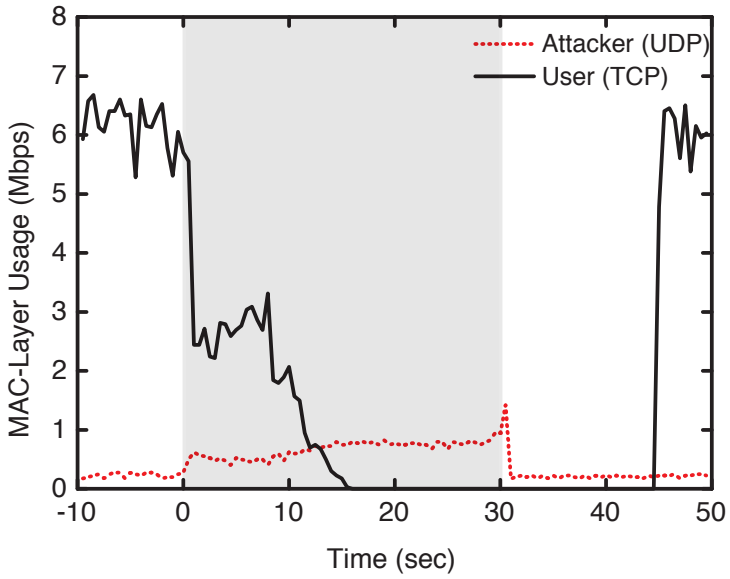
Time fairness can be enforced at the access point by implementing a Time-Based Regulator (TBR) that times each transmission: if user A is allocated time duration t_n in the n^{th} round, then all other users are allocated the same time duration.

To implement a TBR, I use HostAP on a Pentium-III 1 GHz laptop running Linux 2.6.24 so as to gain control of the queuing behavior. The Pentium-III laptop has an Ethernet interface and an Atheros 802.11a/b/g card. In particular, I implement a priority queue at the access point in order to select the next client to serve. I also emulate the rate adaptation algorithm of the Linksys WRT54G access point in order to obtain consistent comparisons of the data rates between the attack scenarios and the implemented mitigation.

I first consider the case where a normal UDP user shares the wireless link with a partial deafness attacker. Both the partial deafness attacker and the normal user have a UDP source rate of 11 Mbps. The partial deafness attacker is configured to only acknowledge the seventh transmission of every packet. The resulting throughput is shown in Table 3.4. When there is no attacker, the user can receive 6.07 Mbps of traffic, which is consistent with



(a) Linksys WRT54GC



(b) Trendnet TEW-432BRP

Figure 3.6: MAC-layer utilization by TCP under the partial deafness attack. The shaded region (0-30 sec) shows the time of attack.

Table 3.4: UDP throughput of normal user and partial deafness attacker with Time-Based Regulator (TBR). The source rate of attacker and normal user is 11 Mbps. Results are averaged over 20 runs.

	Attacker	Normal user
Normal user only		6.07 (Mbps)
Without TBR	110.9 (kbps)	107.9 (kbps)
With TBR	52.5 (kbps)	2.93 (Mbps)

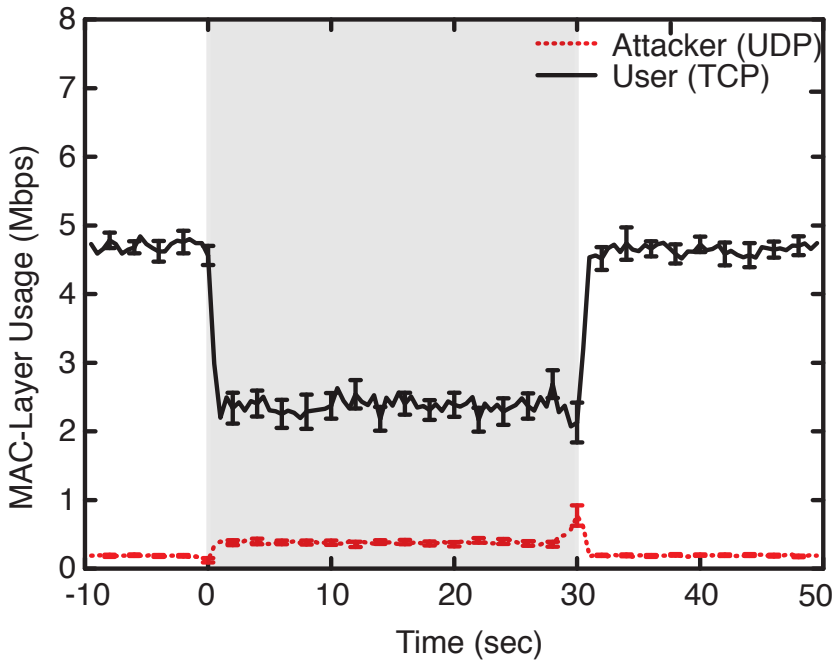


Figure 3.7: The TCP user's MAC-layer channel utilization with the countermeasures. The shaded region (0-30 sec) shows the time of attack; results are averaged over 20 runs, with the error bars (95% confidence interval).

the previous result in Table 3.2. Moreover, when the attacker is present, the user still enjoys almost half of this rate, at 2.93 Mbps, which shows a significant improvement over using access opportunity fairness.

I apply a TBR to a TCP user in the presence of a partial deafness attacker who uses UDP at the transport layer. Figure 3.7 shows that a TBR allows the normal user to obtain significantly better service when under attack. In particular, the normal TCP user ceases to experience heavy packet losses when a TBR is deployed at the access point.

Time fairness can be implemented with 802.11e by choosing an appropriate traffic category for each node according to their fair share of channel

occupancy time [32]. However, 802.11e itself (i.e. 802.11e without TBR) might not be effective as a countermeasure since 802.11e specifies only four traffic categories (i.e. four queues). As multiple partial deafness attackers can connect to a single access point, the attackers can collectively block all four queues used by 802.11e.

3.6 Chapter Summary

In this chapter, I present the *partial deafness* attack that exploits the fairness mechanism of the IEEE 802.11 standard. My attack targets the 802.11 MAC protocol without modifying the MAC-layer implementation. Furthermore, the partial deafness attack does not require the attacker to have better resources than a normal user; the attacker can have lower signal strength, slower computation, and be farther from the base station and still negatively impact the normal users. In particular, the partial deafness attack uses low bit rate and retransmission, two mechanisms that do not affect fair access-opportunity, but creates head-of-queue blocking at the access point.

I show that the partial deafness attack substantially degrades the performance of normal users that use UDP and can almost completely deny service to users using TCP. I then propose and evaluate a time-fair countermeasure that mitigates the partial deafness attack using a time-based regulator. We experimentally show that the countermeasure restores a reasonable level of performance for normal users.

CHAPTER 4

CRAFT: A SECURE TRANSPORT LAYER CONGESTION CONTROL PROTOCOL

A *link* is a communication channel connecting two or more communicating end hosts. A pair of end hosts are connected if between them there exists a path consisting of one or more links. For example, a wireless medium is a link that allows multiple nearby wireless devices to connect to each other; and two non-neighboring end hosts sharing a common neighbor can be connected using a two-link path via the common neighbor. The link with the minimum communication bandwidth on the path between two end hosts is the *bottleneck link*, and the communication throughput between the two end hosts is determined by the bandwidth of the bottleneck link.

In an ad hoc network, any link in the network is a potential bottleneck link since all links are established between end hosts. In a network with infrastructure, however, some links might not be bottleneck links since benign access points only generate negligible traffic compared to end hosts. Moreover, it is commonly assumed that the core of today's Internet infrastructure is over-provisioned and all bottleneck links are topologically close to the end hosts.

If an end host directs to a second end host a large number of data packets, the packets may occupy an unfair amount of bandwidth. In the extreme case, the number of data packets is so large that the packets exhaust all available bandwidth of the bottleneck link between the two end hosts. Any other end hosts sharing the bottleneck link will no longer be able to use it. This type of denial-of-service is known as *flooding*.

Flooding can be both the result of a flash event or a malicious act. In this chapter, I do not try to make a distinction between these two cases. My approach in providing availability is to use fairness enforcement to defend against flooding: We enforce that all end hosts fairly share the communication medium, so that even if a link is congested, all end hosts sharing that

link suffer fairly. We adopt this approach to avoid the undesirable scenario of mistaking a flash crowd as malicious denial-of-service attackers.

4.1 TCP Fairness and TCP Congestion Control

As seen from Chapter 3, we must be careful in selecting the fairness metric in order to prevent malicious exploitation. A common fairness notion is "max-min fairness," where the ratio between the maximum and the minimum of a set of values is minimized; a more obscure fairness notion is the "contested garment fairness" that is analyzed in detail by Aumann and Maschler [33]. In designing CRAFT, which stands for Capability-based Regulation of All Flows and Traffic, I choose to offer fairness defined by the predominant congestion control protocol today: the Transport-Control-Protocol (TCP) fairness. Specifically, I choose TCP congestion control as an *Internet-fair* rate selection algorithm in order to maintain compatibility with legacy TCP traffic.

I briefly introduce the TCP congestion control in order to demonstrate how CRAFT enables each deploying router to enforce that each flow traversing that router follows the TCP congestion control, and in turn to enforce that each flow obtains its TCP-fair bandwidth. In a nutshell, TCP congestion control treats the network as a black box, observing the end-to-end delay and loss characteristics and choosing a bit rate compatible with all intermediate links.

TCP provides a reliable stream service, meaning that each byte sent by the sender-side application is received intact and in-order at the receiver-side application. Because the underlying IP network is unreliable and may reorder packets, TCP must retransmit and reorder packets as necessary. To this end, for each direction of a TCP connection, each byte is given a contiguous *sequence number*. Each packet contains the sequence number of the first byte contained in that packet, as well as an *acknowledgement number*, which represents the sequence number of the last contiguously received byte for the opposite-direction traffic.

The main goal of the TCP congestion control is to estimate the available network bandwidth and adjust the offered load accordingly. The intuition of TCP congestion control is that the sender increases the offered load un-

til it determines (usually through packet loss) that an intermediate link is congested, at which point it reduces the offered load. A TCP sender adjusts its offered load by changing its *congestion window* (cwnd), which represents the maximum number of unacknowledged data bytes that the sender releases into the network at a time.

When a TCP sender initially establishes a connection with a TCP receiver, the congestion window is set to the initial window size (usually two maximum packet sizes), ensuring that the TCP sender starts sending at a low rate. In normal operation, the TCP receiver sends an acknowledgment packet for every other data packet sent by the receiver. Each time the sender gets a new acknowledgment number from the receiver, the sender increases the size of its congestion window, allowing the sender to send more data per round trip time, thereby increasing its data rate. The sender uses one of two algorithms to determine the amount by which to increase the congestion window, the selection of which depends on a parameter called the *slow-start threshold* (ssthresh). The initial value of the slow-start threshold is arbitrary, but changes as the flow experiences congestion. When the congestion window is smaller than the slow-start threshold, the sender uses the *slow-start* algorithm, which increases the congestion window size exponentially over time. Once the congestion window exceeds the slow-start threshold, the sender uses the *congestion avoidance* algorithm, which increases the congestion window size linearly over time.

A sender uses packet loss as a proxy for estimating congestion. A sender can detect packet loss in two ways. One is using the *fast-retransmission* mechanism, and another is to timeout when no new acknowledgment numbers arrive for a period of time.

In fast-retransmission, the receiver reacts to an early out-of-order packet (that is, one with sequence number greater than the next contiguous sequence number) by sending an acknowledgment which will have the same acknowledgment number as the previous acknowledgment packet, known as a *duplicate acknowledgment* (dup-ack). When the sender receives a third duplicate acknowledgment, it considers the following packet to have been lost, and transmits that packet in a process known as *fast-retransmit*. Once this packet is acknowledged, the sender performs *fast-recovery* by halving the size of its congestion window, and setting the slow-start threshold to the new congestion window, thus entering congestion avoidance.

When a sender receives no new acknowledgement for a time exceeding the timeout limit, the sender's congestion control algorithm declares a timeout, reducing the size of the congestion window to one full-sized segment and halves the slow-start threshold. We define the rate equaling two full-sized segments every timeout limit as the *TCP timeout rate*, which indicates the minimum rate an end host can inject data into the network under heavy congestion and packet losses when using the TCP congestion control.

4.2 Related Work

In this section, I overview prior work on the area of reactive DoS countermeasures, capability-based systems, and bandwidth management schemes.

Reactive Countermeasures Reactive countermeasures only react when the network determines that it is under attack. Networks can use anomaly detection mechanisms [34, 35, 36] to detect attacks. After detecting an attack, a defense system can use traceback mechanisms [37, 38] to reconstruct network paths to the source of attack traffic and subsequently react to traffic from such source. In particular, pushback [39, 40] mechanisms provide a way to rate-limit attack traffic. These reactive defense mechanisms have two fundamental problems. First, they need to solve a difficult inference problem to detect an attack before they can react to the attack, and could potentially result in false positives. Second, if the time taken to react and limit the attack traffic is long, the resulting damage can be severe before the system can mitigate the attack.

Filter-Based Systems Filter-based systems partition incoming flows into attack and non-attack flows, and install filters close to the source to block attack flows [41, 42]. When filtering is performed at the end host, filter-based systems cannot stop colluding attackers from flooding a link, because attacking flows never terminate at a legitimate host. Filtering at an intermediate host other than the end-host deprives the filter of application-specific information available only at the end host. Furthermore, filters are unlikely to catch all misbehaving flows, so some mechanism to ensure fairness among unfiltered flows may still be necessary. I therefore consider filter-based systems

to be orthogonal to my approach, which is an improvement on capability-based fairness-enforcing systems.

Capability-Based Systems Capability-based systems proactively prevent attacks from happening instead of mitigating the attacks after the fact. Anderson et al. suggest the application of capability to defend bandwidth exhaustion attacks [43]. Yaar et al. propose the SIFF algorithm [44], and Yang et al. propose the TVA algorithm [45] that are concrete designs of capability-based DoS defense systems. Liu et al. compared the performance of various capability-based systems and filter-based systems [42].

A common problem in capability-based systems is the *denial-of-capability attack*, where an attacker floods a link with connection initialization packets with the intent of exhausting the capability request channel. In previous literature, Parno et al. propose the Portcullis algorithm to defend against this attack [46] by prioritizing service requests based on the computational power used to generate them. The request channel exhaustion attack is orthogonal to the CRAFT protocol, and CRAFT uses Portcullis to prevent the denial-of-capability attack.

Previously proposed capability-based protocols could provide incremental-deployment properties if the congestion point could be determined in advance, and if the autonomous system controlling that point deployed the capability system; Yang et al. first suggested this technique [45]. My work can be seen as the first capability-based system that can be incrementally deployed securely without assuming the knowledge of the congestion point, and in fact without necessarily requiring deployment in every autonomous system.

Bandwidth Management Schemes Fair bandwidth allocation has been studied in depth. Stoica et al. propose a core-stateless fair queuing (CSFQ) algorithm to reduce flow state in core routers [47]. In their proposed CSFQ algorithm, only edge routers perform flow-specific operations. CRAFT can also be used to provide the core-stateless property of CSFQ; moreover, since CRAFT can protect all downstream links, CRAFT can allocate a TCP-fair rate without any help from core routers, unlike CSFQ, in which core routers must process a special CSFQ header. That is, CRAFT can provide CSFQ at a lower cost in the core at the expense of higher costs at the edges.

TCP trunking systems provide aggregated flow fairness. Kung et al. propose a TCP trunking algorithm that emulates the TCP congestion control mechanism; however, this emulation is intended to provide fair rates for the aggregate of flows between edge routers [48], not to provide fair rates for individual flows. CRAFT provides better granularity; each link is shared fairly by all *flows* traversing it. While the trunking mechanism can only guarantee fair sharing in links between two deployed routers, a single CRAFT router can securely guarantee fair sharing on all downstream links.

4.3 CRAFT Design

In this section I present the details of CRAFT and how a CRAFT router provides TCP-fairness for all its downstream links by preventing a CRAFT sender from exhausting the bandwidth of any downstream links. In particular, I explain in detail how CRAFT reacts to packet reordering and loss. For simplicity, I present my proposed protocol as it would operate when data flows in only one direction, even though the CRAFT protocol design, simulation, and implementation extend to bidirectional flows.

4.3.1 CRAFT High-Level Design

CRAFT ensures that all flows adhere to the TCP congestion control specification by emulating the state of each flow at each CRAFT router. If each packet belonging to a particular flow traverses a CRAFT router, the CRAFT router can calculate the maximum allowable size of the congestion window (cwnd) by using the standardized congestion control specifications [49].

Since flows may follow asymmetric routes [50], a single router might not capture both a packet and its acknowledgment. Furthermore, acknowledgments in TCP are not cryptographically dependent on the data in the TCP packet, allowing a receiver to send acknowledgments without actually receiving a packet [51]. A CRAFT router thus controls each CRAFT flow by verifying that a secure token was issued and subsequently received in the following rounds by itself. In other words, a CRAFT router can monitor a flow as long as the path of that flow remains the same.

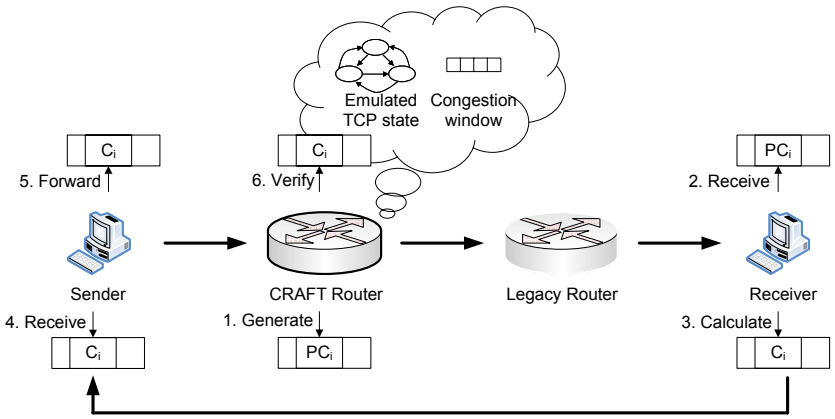


Figure 4.1: Capability-based enforcement of TCP congestion control algorithm. The CRAFT router generates a pre-capability for a flow. After the receiver gets the pre-capability, the receiver calculates and forwards a new capability to the sender. The sender includes received capability to the CRAFT router in a future packet. By verifying the capability, the CRAFT router can know that the previous data packet and acknowledgment were successfully received by the receiver and sender, respectively.

To allow each CRAFT router to monitor the remaining path of the flow from the router to the receiver, a CRAFT router inserts a pseudorandom token in each CRAFT packet routed through that router. I refer to this token as a *pre-capability*. Like SYN cookies, a CRAFT router assumes privacy of plaintext traffic between routers. A CRAFT receiver combines pre-capabilities from each of the routers into a capability for each router; each of these capabilities is then concatenated to form a capability for the path.

A receiver that successfully received the capability issued by each router is likely to also have received the entire packet; thus the capability associated with each CRAFT router can prove to the issuing CRAFT router the delivery of a packet and its acknowledgement. By requiring the sender to include the capability with future packets, a CRAFT router can verify that the receiver has received the previous data packet and the sender has received a legitimate acknowledgement.

Figure 4.1 illustrates how CRAFT generates and maintains capabilities. When a sender initiates a CRAFT flow, it sends a CRAFT request packet, and the router allocates state in its memory corresponding to the new flow, and includes the initial capability for that new flow in the packet. When the flow initiation packet reaches the receiver, the receiver allocates state for the new flow, stores the initial capability value, and sends an acknowledgement

to the sender to finish connection establishment, including in its response the capability corresponding to each router.

Once a sender establishes a connection, the sender allocates a unique contiguous number, the Packet ID, for each packet it wishes to send. The CRAFT Packet ID will be different from, and is somewhat uncorrelated to, a TCP sequence number since the TCP sequence number is issued based on the byte number, while CRAFT Packet ID is issued based on the packet number. To prove to a CRAFT router that the sender has indeed received an acknowledgement, the sender also includes the most recent capability it has received from the receiver. The router first checks to see if this router has previously seen this packet ID for this flow, for example by using a data structure similar to the one IPSec uses for duplicate detection [52]. If the Packet ID is a duplicate, the packet is discarded; otherwise the router checks to see if the most recent capability reflects a new acknowledgement; if so, it processes the acknowledgement as described in Section 4.3.4 and updates the cwnd as described in Section 4.3.5. The CRAFT router then determines whether or not this packet can be sent using the current cwnd; if not, it drops the packet. Otherwise, it replaces the valid capability (which is now no longer needed) with a new pre-capability specific to the Packet ID of this packet and forwards the packet to the next hop.

When the receiver receives a CRAFT packet, the receiver performs a bitwise exclusive-or between the newly received pre-capability $P_{f,i}$ (the pre-capability for Packet ID i in flow f) and the previously stored capability $C_{f,i-1}$ (the capability for Packet ID $i-1$ in flow f) and stores the result of this operation in the new capability, so $C_{f,i} = C_{f,i-1} \oplus P_{f,i}$. The receiver then returns this new capability to the source for use in future packets.

4.3.2 CRAFT Packet Header

In this section, I outline the format of the CRAFT packet header used in the prototype implementation. The header is illustrated in Figure 4.2. This represents one set of design choices, which may not be optimal, but serves to illustrate the data needed in a CRAFT packet header.

In the simulations, CRAFT packets are assigned a specific IP protocol number. When a CRAFT packet carries a data payload, the "Next header"

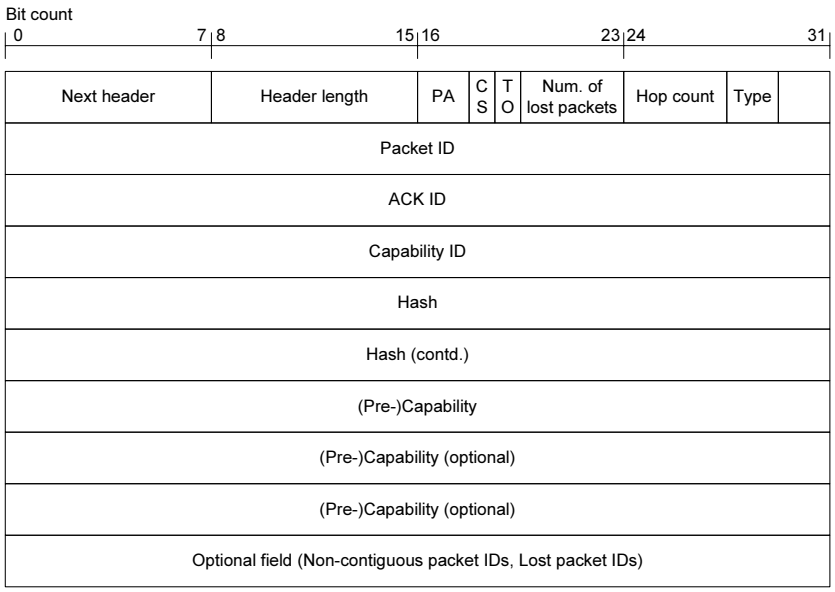


Figure 4.2: Capability packet header format: This header is inserted between IP header and a transport-layer protocol header.

field of the packet carries the protocol number of that data payload, as in IPv6 hop-by-hop and destination options headers. Up to two CRAFT headers can be contained in any packet: the outer CRAFT header (immediately following the IP header) is intended to be visible to and processed by routers along the path, and the internal CRAFT header (following the outer CRAFT header) is used to return capabilities and packet reception data to the receiving host. The "Header length" field represents the number of bytes taken by this entire CRAFT header.

The Partial ACK (PA) field is the number of optional Non-contiguous packet IDs fields at the end of this CRAFT header; these are used to keep track of packets that are received out-of-order. Verifying that these packets were properly received allows a CRAFT router to determine when a sender's initiation of fast-retransmission is authorized.

The Capability Size (CS) field is used to select between short capabilities and long capabilities, as described in Section 4.4.1. The timeout (TO) bit is set when this packet is the first packet following a timeout. The next field indicates the number of lost packets whose Packet IDs are listed in optional fields at the end of this CRAFT header; Lost packets correspond to the packets that would be retransmitted by TCP Congestion Control. The hop count indicates the number of CRAFT-enforcing routers traversed since the

packet left the source. Type indicates whether this packet is an establishment packet or a regular data packet.

Each CRAFT packet is given a contiguous and increasing Packet ID. The ACK ID is the Packet ID *in this direction* of the most recent *contiguously* acknowledged packet (subject to any losses disclosed through Lost Packet IDs). The Capability ID is the Packet ID *in this direction* of the most recent acknowledged packet without any continuity requirement; this is not required for protocol functionality but used to simplify implementation.

The Hash field is a value drawn from a hash chain, and is used to prevent an attacker from spoofing a packet with an arbitrary Packet ID, as described in Section 4.4.2. The sender initializes the Capability/Pre-Capability field to the capability described in this packet header; then each intermediate router changes its part of the capability to a pre-capability for this Packet ID, so that at the receiver, the field contains a pre-capability corresponding to this Packet ID.

Finally the optional fields of non-contiguous packets and lost packets are included in the quantities specified earlier in the packet. Ordinarily the capability is computed over exactly those pre-capabilities corresponding to Packet IDs between 1 and the ACK ID. A non-contiguous packet ID is included when the capability includes a pre-capability from a Packet ID larger than the ACK ID; a Lost Packet ID is used to indicate that the capability omits certain pre-capabilities even though their Packet ID is less than the ACK ID. To prevent the accumulation of lost packet information, once every router has seen a specific Lost Packet ID, it needs not be included in future packets on this flow.

4.3.3 Pre-Capability Construction

Pre-capabilities must be constructed so that any router can efficiently verify its previously issued pre-capabilities; otherwise the router may not be able to handle dense traffic traversing through that router.

Pre-capabilities must also be constructed so that they are *unpredictable*; that is, when a packet is lost, an attacker cannot predict the pre-capability of that packet; otherwise the attacking receiver can acknowledge lost packets and continue to grow its cwnd in the face of congestion.

To provide both properties, we generate pre-capabilities using a cryptographically secure hash function g:

$$P_{f,i} = g(K, f, i),$$

where K is the secret key of the CRAFT router, f is the flow ID, and i is the Packet ID from the CRAFT header. In our implementation, the ID of a flow is a large value randomly generated by the router when the flow is created. If g is a hash function in the Random Oracle Model [53], then $P_{f,i}$ is indistinguishable from a random number because the pair (f, i) has not been previously seen.

When a receiver receives contiguous packets, it combines the pre-capabilities corresponding to those packets using the exclusive-or (xor) function. For the sake of efficiency, we would like to be able to verify a large set of combined contiguous pre-capabilities in constant time. We use a telescoping construction:

$$g(K, f, i) = E_K(f||i) \oplus E_K(f||i+1),$$

where E_K represents a computationally efficient keyed MAC such as HMAC [54]. By choosing E_K as a secure pseudo-random function, it follows that $P_{f,i}$ is also indistinguishable from a random number.

The receiver constructs the capability associated with ACK ID i by computing the exclusive-or of all pre-capabilities up to i,

$$C_{f,i} = P_{f,0} \oplus \ldots \oplus P_{f,i}.$$

By combining pre-capabilities at the receiver using the exclusive-or function, we take advantage of a desirable property where if any input (and its distribution) is unknown, the output is uniformly distributed over the domain, yielding the largest uncertainty and secrecy in the information theoretic sense. Moreover, since $P_{f,i} = E_K(f||i) \oplus E_K(f||i+1)$, the capability can be calculated efficiently using

$$C_{f,i} = E_K(f||0) \oplus E_K(f||i+1).$$

Since a capability is formed by xor'ing all pre-capabilities associated with a (flow, router) pair, the size of a capability is the same as a pre-capability. I

design CRAFT so that each autonomous system deploying CRAFT needs not trust its neighboring autonomous systems; thus, each CRAFT flow should be policed by one CRAFT router in that autonomous system. Consequently, the overhead of CRAFT is directly related to the number of deploying autonomous systems on the path multiplied by the length of each capability. To minimize overhead, the prototype CRAFT implementation uses three-bit "short" capabilities. The security of using short capabilities will be discussed further in Section 4.4.1.

4.3.4 Capability Verification under Packet Loss

As long as no packets are lost, the capability when ACK ID is i can easily be verified using $C_{f,i} = E_K(f||0) \oplus E_K(f||i+1)$. In this section, I discuss how the router can efficiently verify capabilities despite packet losses.

The ACK ID stores the largest *contiguous* packet ID that the receiver is acknowledging, subject to disclosed packet losses. For example, if a sender has received acknowledgments for packets 1 through i, and packet $i+1$ is lost, then the acknowledgment for packet $i+2$ will reflect an ACK ID of i and a partial acknowledgment of $i+2$. The acknowledgment for packet $i+3$ will reflect an ACK ID of i and a partial acknowledgment of $\{i+2, i+3\}$. When packet $i+4$ arrives at the receiver, the receiver knows that this third duplicate acknowledgment would trigger fast-retransmission, so it sends an ACK ID of $i+4$ and a lost packet of $i+1$.

When the sender receives this acknowledgment for packet $i+4$, it will adjust its cwnd in accordance with TCP fast-retransmission; we will discuss cwnd calculations in Section 4.3.5. In its subsequent packets $j, j+1, \ldots$, the sender will include a lost packet field to indicate that packet $i+1$ was lost. Since Packet IDs are used sequentially, the sender does not reuse the Packet ID $i+1$. Instead, when the sender receives an acknowledgment for a Packet ID of at least j, it knows that each router on the path knows about the loss of packet $i+1$, and can also stop including a lost packet field for packet $i+1$ in subsequent packets.

I now present the algorithm that CRAFT routers use to verify capabilities in real network conditions, which can include packet losses and reorderings. My algorithm has constant per-flow storage overhead, regardless of the num-

ber of losses experienced by each flow. For each flow, the router keeps track of the largest ACK ID for which a capability was correctly verified. Temporarily ignoring partial acknowledgements (out-of-order acknowledgements with Packet ID greater than ACK ID), it also keeps track of the most recent capability verified in this way. For example, if ACK ID i were verified without losses, then the router would keep track of $(i, C_{f,i})$.

When a later packet with ACK ID $i+4$ and Lost Packet ID $i+1$ arrives, the router checks to make sure that the presented capability C equals $C_{f,i} \oplus P_{f,i+2} \oplus P_{f,i+3} \oplus P_{f,i+4}$; that is, the new capability is consistent with the last valid capability xor the pre-capabilities between the last valid capability and the current capability. More generally, if the router stores ACK ID i and capability C, and the new packet indicates ACK ID $j > i$, lost packets $\mathbb{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$, and capability C', then C' is valid if and only if $C' = C \bigoplus_{\{x \in ([i+1,j] \setminus \mathbb{L})\}} P_{f,x}$; that is, when we xor C with all pre-capabilities between $i+1$ and j that are not associated with a lost packet, the result should be the new capability.[1] If C' is valid, we update the (ACK ID, Capability) pair to (j, C').

When a packet includes partial acknowledgments, the verifying router can efficiently verify them using the property of the xor function. In particular, if a packet has capability C and partial acknowledgment a_1, a_2, then, for purposes of verification, it is the same as having capability $C \oplus P_{f,a_1} \oplus P_{f,a_2}$ and no partial acknowledgments. The verifying router therefore adjusts for partial acknowledgments before proceeding with the verification steps discussed above. This also means that any stored capabilities do not include partial acknowledgments, minimizing the necessary router state.

4.3.5 Congestion Window Calculation

When a CRAFT router verifies a new capability, it knows that one or more additional packets have been acknowledged by the receiver, so the router updates its emulated cwnd. Because the TCP protocol description leaves some room for interpretation, a flow's exact cwnd depends on the TCP implemen-

[1] Since $x \oplus x = 0$, the router can efficiently calculate

$$C \bigoplus_{\{x \in ([i+1,j] \setminus \mathbb{L})\}} P_{f,x} = C \oplus E_K(f||i+1) \oplus E_K(f||j+1) \bigoplus_{x \in ([i+1,j] \cap \mathbb{L})} P_{f,x}.$$

tation. I thus only provide a reasonable bound on the cwnd according to the standard [49].

Like a TCP sender, a CRAFT router maintains state on the cwnd, slow-start threshold (ssthresh), and maximum segment size of each flow. The sender discloses the maximum segment size at connection initialization, and whenever the maximum segment size changes. The CRAFT router updates this state in a manner consistent with the behavior of a legitimate TCP sender-receiver pair.

The CRAFT router can distinguish between acknowledgments generated from contiguous packets and those generated due to out-of-order delivery by examining the ACK ID and the partial acknowledgment fields. The cwnd and ssthresh are initialized to some predefined values suggested by the TCP standard. In the case of contiguous packets, the CRAFT router emulates a sender that has received one acknowledgment per packet, which is the maximum that the TCP standard allows the receiver to send. Though a unidirectional TCP flow uses delayed acknowledgments, i.e. sending only one acknowledgment for every two received packets, a bidirectional TCP flow opportunistically carries acknowledgments on new reverse traffic, allowing the possibility of one acknowledgment per packet. Since the cwnd grows faster in this latter case, we choose to emulate the cwnd according to one ACK per packet in order to form a reasonable upper bound of the cwnd.

When a CRAFT router determines that acknowledgments are generated due to out-of-order delivery, it does not adjust the cwnd until three such duplicate acknowledgments are detected, at which point the sender must include a Lost packet ID in the CRAFT header. The CRAFT router then emulates the TCP state associated with fast-retransmission and fast-recovery. Because a CRAFT router knows specifically which packets were lost and which packets were transmitted in response to that loss, it can easily detect fast-retransmission and emulate fast-recovery.

Specifically, when the CRAFT router sees a Lost Packet ID, the router-emulated flow state enters fast recovery phase. The CRAFT router reduces the ssthresh to the maximum of half of the amount of outstanding packets or two times the sender's maximum segment size (SMSS) and sets the emulated cwnd to the new ssthresh plus 3 SMSS to take into consideration the three duplicate acknowledgments. The CRAFT router considers the first packet with disclosed packet loss as a retransmitted packet and records that packet's

Packet ID. When the CRAFT router sees an acknowledgment with increment of 1 during fast recovery phase, it increases the emulated cwnd by 1 SMSS to keep the same number of outstanding packets. When the CRAFT router sees a packet with an ACK ID equal to or larger than the recorded ID of the retransmission packet, the emulated state exits from fast recovery phase and the CRAFT router sets the emulated cwnd to ssthresh.

In some circumstances, a TCP sender loses a packet and cannot recover through the fast-retransmission mechanism. In these cases, after a certain period of time, the sender times out. To reflect recovering from a timeout in CRAFT, the timeout (TO) field is set to 1. A router that sees a packet with the timeout bit set to 1 then accepts any capability borne by this packet (to reduce the burden of listing all lost packet IDs), resets the cwnd to one packet, and halves the ssthresh.

4.3.6 Other Details

Similarly to other proposed capability-based systems, a CRAFT router allocates a certain amount of bandwidth to legacy traffic depending on its admission control policy, so that users who have not deployed CRAFT can still have access to network service. Since users can increase their traffic rate arbitrarily, a CRAFT router limits the bandwidth allocated to legacy traffic by fair-queuing.

If the route used by a CRAFT flow changes, any new CRAFT router on the path will detect that this CRAFT flow is not associated with a previously established connection. Any such router will ignore the CRAFT header in the packet, treat the packet as legacy traffic, and send a reset message to the corresponding sender of such flow. The sender can then reinitiate a new CRAFT flow to the destination.

4.4 Security Analysis of CRAFT

In this section, I discuss several possible attacks on the basic design of CRAFT as explained in Section 4.3 and describe my proposed defense mechanisms.

4.4.1 Capability Forgery Attack

A CRAFT router disregards all packets with invalid capabilities. That is, if a flow wishes to increase its rate above the CRAFT rate, it needs to send more capabilities than it receives. An attacker can guess future acknowledgments by flooding the network with random capabilities. Only $1/(2^3) = 1/8$ of which are expected to be accepted by the first CRAFT router on the path. If there are n deploying CRAFT routers on the path between the attacker and the bottleneck link, then $1/(2^{3n}) = 1/(8^n)$ of the packets are expected to enter the bottleneck link. Since deploying CRAFT routers may be sparse in the initial phase of deployment, there may not be enough CRAFT routers on the path to prevent exhausting the bottleneck link bandwidth.

Therefore, when a CRAFT router receives an excessive number of incorrect capabilities from a particular flow, it notifies the sender to switch to using long capabilities instead of the 3-bit short capabilities used under normal operations. We choose this approach rather than penalizing the flow to prevent the attack where an attacker alters the capability of a packet from a legitimate flow, thereby harming the performance of the benign flow.

When using long capabilities, such as those 32-bits long, the probability of guessing a valid capability is vanishingly small, and an attacker that tries to guess random capabilities will have its bandwidth reduced by a factor of 2^{32} at each legitimate CRAFT router. To put this in perspective, an attacker flooding an OC-768 link at line-speed can only squeeze 9.3 bps past the first CRAFT router. Each additional CRAFT router provides a further reduction in bandwidth. A router determines whether short or long capabilities are in use using the Capability Size (CS) field of the CRAFT header.

4.4.2 The Spoofing Attack

A CRAFT router removes duplicate packets that share a Packet ID within a single flow. Thus, an attacker that forges a packet with Packet ID p from a benign flow causes the actual benign packet bearing Packet ID p to be dropped by the CRAFT router.

I propose using a 64-bit hash chain to authenticate the Packet ID. A hash chain consists of a series of hash values generated by a cryptographic hash function [55]. A cryptographic hash function takes a message of arbitrary

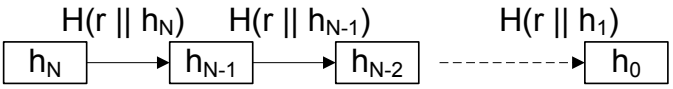


Figure 4.3: Generation of a hash chain

length as input, and computes a fixed-length output, called a message digest. Such a function is *preimage resistant* if given $h = H(m)$, it is infeasible to obtain m.

To generate a hash chain of length N, the sender chooses two nonces, r and h_N, and generates the chain by feeding back the output of a preimage resistant hash function as the input. That is, $h_{i-1} = H(r||h_i)$, where the $||$ operation denotes concatenation. Figure 4.3 illustrates the generation of a hash chain of length N. The nonce r is included in the input to avoid reusing a hash value of other flows.

To allow each CRAFT router to authenticate Packet IDs, a sender includes in its CRAFT connection initiation packet the values of r and h_0. Each CRAFT router stores the values r and h_0 to authenticate future hash values. When the sender sends Packet ID m, it includes the hash value h_m in the hash field of the CRAFT header.

When a CRAFT router receives a new packet, it can authenticate the Packet ID by traversing the hash chain. For example, when the CRAFT router receives Packet ID 1 with hash element h', it will verify that $H(r||h') = h_0$, and if so, the CRAFT router will know that $h_1 = h'$. Since the cryptographic hash function is preimage resistant, it is infeasible for other users to generate h_i from h_{i-1} in a reasonable amount of time. Thus, a CRAFT router can mitigate the spoofing attack by authenticating Packet IDs with minimal effort when the router decides that authentication is required.

4.4.3 Compromised Router

If a router is compromised, then it can arbitrarily drop packets traversing that router. This problem is orthogonal to capability-based systems for preventing bandwidth exhaustion.

The router can also collude with an endpoint, but this provides no advantage over using the router as an end point directly. For example, if the compromised router R is on the path between malicious nodes M_1 and M_2, R

can facilitate their bandwidth overuse as well as if M_1 establishes a CRAFT connection to R and M_2 establishes a CRAFT connection to R, and both of these connections are used to their full capacity.

4.4.4 Memory Exhaustion Attack

Since a CRAFT router needs to maintain some state in memory for each flow, an attacker can attempt to exhaust the memory space of the router by establishing an excessive number of connections. CRAFT can use Portcullis [46] to mitigate this attack by allocating a per-computation-fair share of the available number of connections among all requesting end hosts.

A CRAFT router can also dynamically adjust the difficulty of the computation puzzle as a function of its free memory. Specifically, when the memory of a CRAFT router is mostly unused, the router uses easier computation puzzles to encourage more flows to use CRAFT; it otherwise uses computation puzzles that are more difficult to mitigate any possible memory exhaustion attack.

An attacker can also attempt to exhaust the memory space of each packet. In each CRAFT packet header, a limited memory space is allocated for pre-capabilities. Thus, if an attacker fills that limited memory space with fake pre-capabilities, no CRAFT routers can monitor the flow since no CRAFT routers can insert pre-capabilities.

A CRAFT router mitigates this attacker by assigning flows to use the legacy portion of the link controlled by the router. When flows use 3-bit short capabilities, the CRAFT packet header can support 16 CRAFT routers monitoring the flow. Since we assume routers can trust other routers in the same autonomous system, the CRAFT packet header allows a flow to traverse 16 autonomous systems. A flow that traverses a path containing more than 16 autonomous systems is not likely to have a high rate due to the length of such a path; thus assigning such flows to compete with other legacy traffic allows CRAFT to better serve the high-priority flows.

4.4.5 Imperfections in TCP Feedback

Because TCP measures congestion window according to bytes acknowledged and not the total number of bytes outstanding, and because packet sizes can vary, an attacker could exploit these differences to temporarily overwhelm a link protected by CRAFT. For example, after an attacker acquires a sizable congestion window, he can send multiple 1-byte packets. Since each 1-byte packet comes with its own IP and CRAFT header, an attacker can easily flood the network with traffic that is magnitudes larger than allowed by his congestion window.

In order to avoid these short-term attacks, CRAFT enforces certain properties that TCP does not require. In one approach, CRAFT can enforce Nagle's algorithm [56]; that is, a CRAFT router allows each flow to have one outstanding packet that is of size less than the maximum segment size. Enforcing Nagle's algorithm enables a CRAFT router to prevent an attacker from sending excessively many small-sized packets.

In a more complex but application-friendly approach, when an application needs to send a large number of small packets, the sender notifies all routers each time it changes the maximum segment size. When a node increases its maximum segment size, the congestion window can remain unchanged, because the total number of bytes that can be sent for a given congestion window decreases as the maximum segment size increases. This is because as the maximum segment size increases, the number of packets that can be sent for a given congestion window decreases; thus the amount of overhead associated with those packets also decreases.

Conversely, when decreasing the maximum segment size, the amount of bandwidth associated with a given congestion window increases. If the sender can decrease its maximum segment size without also decreasing its congestion window, then the attacker can temporarily send at much greater than its fair rate. Thus, when a flow *decreases* its maximum segment size, each CRAFT router adjusts that flow's congestion window (cwnd) to ensure that its old maximum data-plus-header in flight is equal to its new maximum data-plus-header in flight, by satisfying the expression

$$\mathrm{cwnd}_{\mathrm{old}} + \eta \frac{\mathrm{cwnd}_{\mathrm{old}}}{\mathrm{SMSS}_{\mathrm{old}}} = \mathrm{cwnd}_{\mathrm{new}} + \eta \frac{\mathrm{cwnd}_{\mathrm{new}}}{\mathrm{SMSS}_{\mathrm{new}}},$$

where SMSS is the sender's maximum segment size, and η is the header length.

4.5 Evaluation

To evaluate the effectiveness of CRAFT and compare it to other DoS defense mechanisms, I implement CRAFT in a Linux kernel and in a network simulator. This section describes the experimental methodology and results. In particular, I implement CRAFT to verify that the design is TCP-friendly, and to analyze the overhead associated with CRAFT; moreover, I simulate a partial deployment scenario and quantify the effectiveness of CRAFT and other DoS defense mechanisms.

4.5.1 Experiment Methodology

The experiment testbed consists of a small network in the Coordinated Science Laboratory (CSL) at the University of Illinois at Urbana-Champaign that is connected over the Internet to Emulab,[2] a public open network testbed, as illustrated in Figure 4.4. The small access network in CSL consists of two senders connected to one router, which is in turn connected to the Internet. The receiver in Emulab is also connected to the Internet. Each of the two senders sends TCP or UDP packets to the receiver in Emulab over the Internet. I place the Internet between the senders and the receiver to experience realistic cross-traffic and queuing delay. The link bandwidth between the senders and the router is 100 Mbps, and I vary the receiver's access link from 10 Mbps to 100 Mbps.

I implement CRAFT between the network layers, including header insertion, processing, and extraction steps at the senders, receiver, and router. This implementation was a Linuxkernel-level implementation so I could most efficiently handle any CRAFT header calculations.

[2]http://www.emulab.org

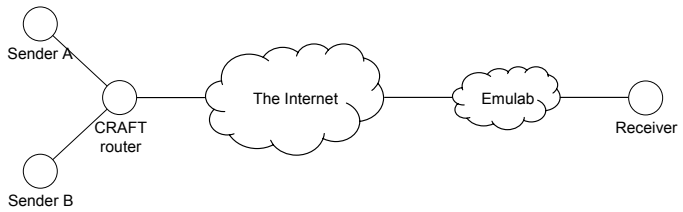


Figure 4.4: Experimental setup to evaluate whether CRAFT is TCP-friendly

4.5.2 Fairness Measured by Implementation

The main design goal of CRAFT is to let flows share link bandwidth in a TCP-fair manner. In this dissertation, TCP-fairness does not mean that traffic rate of a flow under CRAFT exactly matches the rate of a normal TCP flow, because a CRAFT router's emulated cwnd determines not the exact cwnd, but the upper bound of the cwnd allowed by the TCP congestion control.

I first examine how a CRAFT UDP flow shares bandwidth with other TCP flows. Let sender A, a group of one to five TCP senders, send packets to a receiver using Reno TCP in Linux kernel version 2.6.20. Then let sender B, a single UDP sender with 10 Mbps source rate, send packets using CRAFT. That is, B generates a 10 Mbps UDP traffic, but only transmits data allowed by CRAFT. All flows from A and B converge at a CRAFT router before entering the Internet and are eventually delivered as shown in Figure 4.4. In order to make a unidirectional UDP flow compatible with CRAFT, I implement a user-level program to send CRAFT acknowledgments in the reverse direction.

I measure the throughput of B's CRAFT-rate flow and the aggregate throughput of A's TCP flows while varying the number of TCP flows sharing the access link from 1 to 5. Figure 4.5 shows our results. The thin solid line represents the bandwidth that each flow would get if all flows equally shared the link bandwidth. The bold solid line shows the throughput of the CRAFT flow. We observe that the CRAFT flow enjoys 20% to 55% higher throughput than its fair share.

The difference between CRAFT-rate and fair share can be attributed to TCP flows using *delayed acknowledgments*. That is, the TCP standard recommends sending one acknowledgment for every other received packet with some timing criteria; however, CRAFT bounds the maximum TCP rate; thus

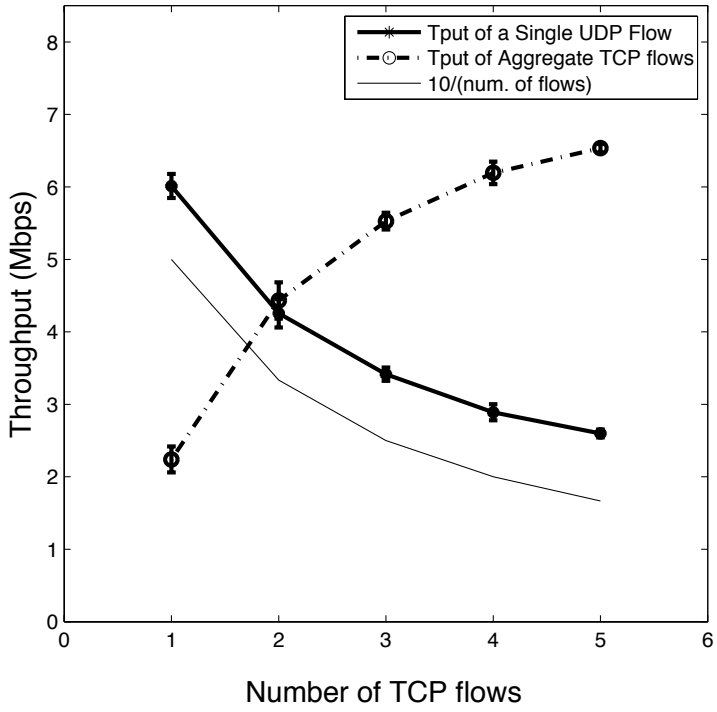


Figure 4.5: Throughput fairness as evaluated by implementation. A single UDP flow, controlled by CRAFT, competes against one to five legacy TCP flows.

CRAFT does not use delayed acknowledgments. Since CRAFT acknowledges packet reception more often than regular TCP flows, the cwnd of a CRAFT flow is allowed to grow faster than a regular TCP flow, but only by a fixed amount.

4.5.3 Overhead Measured by Implementation

To enforce TCP congestion control on each flow, CRAFT incurs two types of overhead: packet header overhead and processing overhead. Packet header overhead decreases the amount of goodput with a given maximum segment size, and processing overhead increases the time a router takes to forward a packet.

Packet Header Overhead The CRAFT header takes additional space in the packet, thus reducing the maximum amount of data sent in each packet. The reduced goodput can be calculated theoretically as

$$\frac{(\text{MSS} - \eta_{\text{CRAFT}})}{\text{MSS}} \times \text{goodput}_{\text{original}},$$

where MSS is the maximum segment size, η_{CRAFT} is the size of the CRAFT header, and $\text{goodput}_{\text{original}}$ is the goodput of a flow without CRAFT deployment. Since the size of a CRAFT header varies due to various optional fields, such as Non-contiguous packet ID's and Lost packet ID's, I do not calculate a theoretic packet overhead, but instead determine it experimentally.

To determine the overhead, I consider a TCP sender sending traffic through the Internet and Emulab to the receiver. We determine the overhead by comparing the goodput of the TCP flow when CRAFT is in use to that of a normal TCP flow. I vary the access link bandwidth between 10 Mbps and 100 Mbps, and perform five runs for every data point. One set of data points reflects legacy TCP in a vanilla Linux kernel, and one set of data points reflects the performance of TCP with CRAFT. When measuring the goodput, I subtract the CRAFT header from the packet size in order to determine the goodput of the TCP data and headers. Figure 4.6 shows the mean TCP goodput for each access link rate and the 95% confidence intervals of the average. These results show that the legacy TCP flow provides 2.03% to 15.33% higher goodput than that of CRAFT.

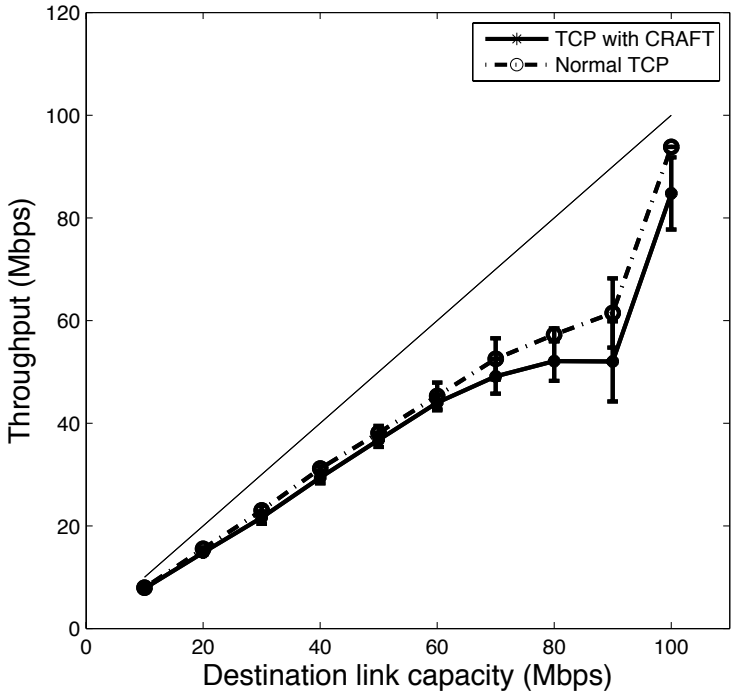


Figure 4.6: CRAFT overhead results

When the access link bandwidth is under 60 Mbps, the difference between the goodputs of the two flows is not significant. The biggest difference between goodputs is observed when access link bandwidth equals 90 Mbps. The goodputs of both CRAFT and normal TCP level off when the access link bandwidth is between 60 to 90 Mbps. We believe that the reduction in bandwidth is caused by traffic shaping somewhere outside of the experiment control and is irrelevant to the CRAFT overhead.

Processing Overhead Upon receiving a CRAFT packet, a CRAFT router performs several operations: looking up the flow, checking the validity of the hash, verifying the capability, updating the congestion window, and generating a new pre-capability. I implement these functions and emulated a CRAFT router using one core of a 3 GHz Xeon processor. I use SHA256 to generate and verify the hash chain, and RC5 to generate pre-capabilities. Table 4.1 shows the measured processing times for each functional block.

My results show that a single core of a Xeon processor can handle over 300,000 packets per second. At a minimum packet size of a 20 byte IP

Table 4.1: Processing times for router functions

Function	Processing time
Creation of a new flow	3120 ns
Lookup of flow	30 ns
Calculation of hash	1720 ns
Verification of capability	300 ns
Update of *cwnd*	170 ns
Calculation of pre-capability	610 ns

header and 28 byte CRAFT header, this represents a bandwidth exceeding 14.4 Mbps. At a more realistic packet size of 1500 bytes, this represents 450 Mbps. Thus low-bandwidth routers can use a slow processor since the processing requirement is low, and a high-bandwidth router can implement these functions to process even more quickly on a FPGA or ASIC, easily obtaining better performance than the Xeon CPU. These numbers are for our particular implementation of CRAFT, and might be improved in future implementations, so they serve as a lower bound for CRAFT performance.

4.5.4 Simulation Methodology

To evaluate the effectiveness of CRAFT in providing fairness, I measure the effect of a DoS attack in a network with only partial deployment. Although the implementation shows that a CRAFT-controlled flow competes fairly with legacy flows, the best way to understand the differences between CRAFT and other previously proposed capability-based DoS defense schemes is to launch a DoS attack. I choose to implement this attack through simulation. I use the ns-2 simulator[3] and construct a network topology that reflects the topological properties of the Internet. I compare the performance between SIFF [44], TVA [45], and CRAFT. I do not consider filter-based systems [41, 42] and the denial-of-capability defense system, Portcullis [46], since they are orthogonal to CRAFT. To simulate the performance of SIFF and TVA, I use the simulation package published by the authors of StopIt [42] and TVA [45].

There is no standard methodology for evaluating DoS defense mechanisms. I thus follow the intended methodology used in the recent work of

[3]http://www.isi.edu/nsnam/ns

Liu et al. [42]. My intended simulation scenario includes 1,000 to 10,000,000 compromised hosts sending traffic at the rate of 10 kbps each. All traffic traverses a bottleneck link that has a 1 Gbps capacity. Due to memory constraints, I adopt a 1/200 scale in the simulation while maintaining the ratio between the aggregate attack traffic and bottleneck link. Namely, the actual simulation is done using 1 to 5000 compromised hosts each sending traffic at a constant rate of 10 kbps, and the capacity of the bottleneck link is subsequently reduced to 500 kbps. My simulation topology is shown in Figure 4.7. Since each compromised host is sending 10 kbps of constant-rate traffic, the set of compromised hosts can be considered one powerful attacker that can send traffic using the aggregate constant rate.

I set the bottleneck link as a legacy link that does not support SIFF, TVA, or CRAFT. This legacy link may, for example, exist in an autonomous system that has not yet deployed a capability system. Because the legacy link does not support the capability system, when the legacy link can no longer sustain the traffic sent by the attackers, it will start dropping packets. I then simulate ten legitimate TCP end hosts each trying to send 100 files, each of size 20 kByte. Each sender sends one file at a time, after the delivery of the previous file is complete. A file delivery is considered complete either when the entire file is successfully transferred or after the file experiences eight timeouts, at which point the file transfer is considered to have failed with transfer time of 1000 seconds. I then measure the fraction of file transfers that successfully complete and the average transfer time of the legitimate users' successful file transfers. These metrics represent how well TCP traffic can compete against the attackers when a capability-based DoS defense mechanism is only partially deployed.

To build a realistic topology, I use the IPv4 Routed /24 Topology Dataset[4] constructed from CAIDA scamper probes, a successor of skitter.[5] Scamper probes record the IP addresses of each hop from a source to a destination and round trip times of intermediate hops as well as a final destination. The destinations are randomly collected from the routed IPv4 /24 prefixes in the BGP tables of the RouteViews project.[6] By using the BGP tables to translate IP addresses of intermediate hops into AS numbers, I construct an AS-level

[4]http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml
[5]http://www.caida.org/tools/measurement/skitter
[6]http://www.routeviews.org

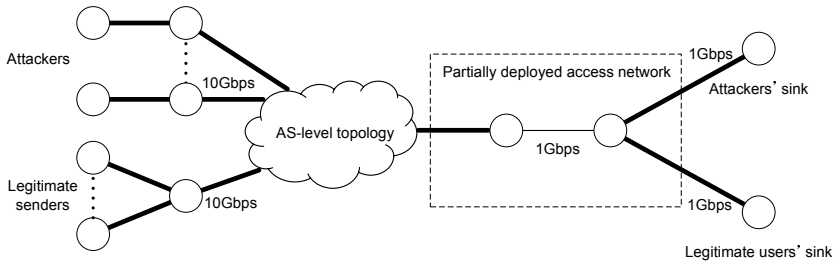
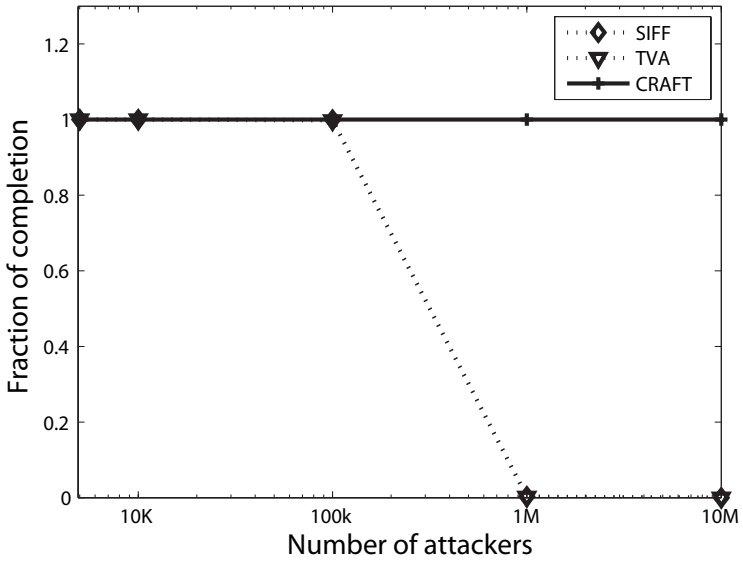


Figure 4.7: Simulation topology: we construct an AS-level topology using a data set of hop-by-hop delay measurements from the Internet. In the access network of receivers, there is a shared link not deploying capability-based systems.

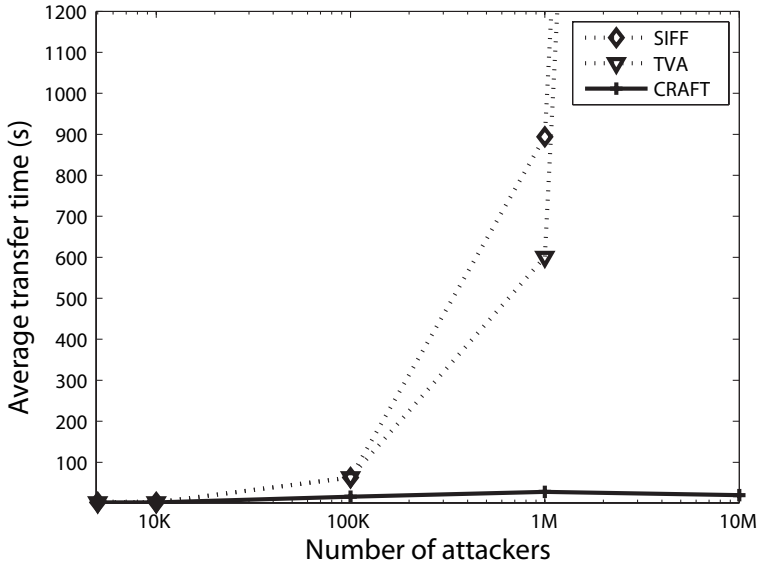
topology in which each link has inter-AS delay. I use an August 2008 dataset measured in San Jose, California by the RouteView project. A topology with delay is necessary to accurately model TCP performance because congestion window evolution depends on round trip time between end hosts [57]. To the author's knowledge, there is no Internet-scale link-bandwidth data, so I set core routers' link bandwidth of our topology at 10 Gbps so that the core routers do not experience congestion. Figure 4.7 shows the simulation topology including access networks, where ten legitimate users are connected to a single common access AS and attackers' access AS's are chosen at random. There exists a legacy link that does not deploy capability-based systems; this link is shared by all attackers and legitimate users, and is denoted by the thin segment in the box in Figure 4.7. The thick lines indicate links on which the capability system is deployed.

4.5.5 Simulation Results

Figure 4.8 shows the fraction of file transfer completion and the average transfer time of files. When the number of attackers is less than 100,000 (e.g., the aggregate constant traffic rate is less than 1 Gbps, the capacity of the bottleneck), the aggregate attack traffic is less than the capacity of the legacy link. Hence, all file transfers are successful and the average file transfer times for all evaluated schemes are similar. When there are 100,000 attackers (e.g. the aggregate constant traffic rate is equal to 1 Gbps), the aggregate attack traffic rate matches the bottleneck link bandwidth. All files are still successfully transferred, but due to queuing at the bottleneck link, the



(a) Fraction of file transfer completion



(b) Average transfer time

Figure 4.8: Simulation results

average transfer time increases. Specifically, the CRAFT system experienced a 18-fold increase while SIFF and TVA both experienced a 68-fold increase in the average transfer time, as shown in Figure 4.8(b).

CRAFT significantly outperforms SIFF and TVA in this situation because attackers are limited by the CRAFT rate, thereby allowing legitimate users to complete file transfer faster than users using other DoS-prevention schemes. When the number of attackers exceeds 100,000, almost all file transfer attempts fail under SIFF and TVA, and the average successful transfer time increases sharply. CRAFT, on the other hand, provides perfect file transfer success rates, and the growth rate of the average transfer time is much smaller. A legacy link in a system that partially deploys SIFF and TVA cannot provide fair service even though the flows going through the legacy link come from users deploying SIFF or TVA. In contrast, in a partially deployed CRAFT system, the attack traffic and the legitimate traffic both reduce their traffic rates and share the bottleneck link in a TCP-fair manner.

4.6 Chapter Summary

This chapter presents the CRAFT protocol, which provides security against DoS attackers by enforcing TCP-fairness on all flows that traverse a CRAFT router. CRAFT does not suffer from common weaknesses observed in reactive mechanisms, such as false alarms. Moreover, CRAFT also does not require routers to trust each other.

A single CRAFT router can protect all links behind it; thus CRAFT can be deployed incrementally and provide strong benefits even to very early deploying end hosts. Our simulation showed that in realistic partial-deployment environments, CRAFT can significantly outperform previously suggested capability-based schemes in both packet delivery success rate and average successful transfer time.

CHAPTER 5

JIM-BEAM: JAMMING AND INTERFERENCE MITIGATION USING BEAM-FORMING ANTENNAS AND RANDOMIZED ORIENTATION

In a wireless network, a serious threat to availability is *jamming*, where an attacker injects noise into the frequency band used for communication, preventing the receiver from correctly decoding data bits. The problem of jamming mitigation has recently been heavily explored. In particular, *spread spectrum* is a well-known method to resist interference when a secret (e.g. a spreading code) is shared between a sender and a receiver [58]. Recent work has extended this approach to *broadcast environments*, where the attacker may have the ability to decode messages; these approaches rely on asymmetric computational burdens [59] and asymmetric knowledge [60]. Other coding-based approaches use indelible marks [61] to transmit messages that cannot be erased by a malicious attacker.

Several uncoordinated spread spectrum (USS) techniques have been proposed to address jamming mitigation in networks without any pre-shared keys or spreading codes [59, 62, 63, 64]. A USS system generally has a published set of possible codes, a transmitter sends a message using a spreading code randomly chosen from among the published set of possible codes. Each receiver finally exhaustively searches for the message using the set of possible codes. Since learning the random spreading code enables a jammer to jam a message, USS systems must assume that no jammer can determine the random spreading code of a packet, and within the same packet duration, jam with that information. As long as this constraint on the jammer's computational ability is satisfied, the USS techniques are attractive since receivers and the transmitter do not need to share a key in order to communicate; thus USS can better support dynamic membership than traditional spread spectrum techniques.

In this chapter, I extend the USS techniques by exploiting the orientations of directional antennas, which is another line of traditional anti-jamming

approaches. Compared to prior USS approaches that are based on coding, directionality shows great promise for mitigating the jamming attack: In coding-based approaches, a message is sent on a specific *code*, and an adversary can choose to jam a single randomly selected code (called *narrow-band* jamming), jam a set of randomly selected codes at reduced power (called *partial-band* jamming), or jam the set of all codes at much reduced power (called *wideband* jamming). Several studies, under different channel assumptions, conclude that the optimal proportion of spectrum on which to jam is a function of the signal-to-interference-plus-noise ratio (SINR) [65, 66, 67]. In particular, a powerful jammer's optimal jamming strategy that maximally increases the bit-error rate is wideband jamming.

However, each attacking transmitter can only be in a single place at any instance in time. Thus, in an idealized scenario, a receiver using a sectored antenna is either affected by that transmitter or not; that is, each adversarial transmitter, physically, must act in a narrow-band fashion in the spatial domain. In other words, directional antennas deprive powerful attackers of the most effective attack methodology.

While the use of directional antennas can mitigate wideband jamming in the spatial domain, their use seemingly contradicts the goal of a broadcast message, which tries to reach all nodes within the neighborhood of a transmitter. In this chapter, I explore the concept of using *randomly oriented* directional antennas to mitigate persistent interference from jammers. In particular, I extend previous randomization-based coding strategies to spatial randomness. I let each wireless device be equipped with one or several directional antennas. Each wireless device reorients the main-lobe of one or more of its directional antennas toward different and randomly selected directions at different times. I call my scheme *JIM-Beam*: Jamming and Interference Mitigation using Beam-forming antennas.

JIM-Beam mitigates jamming by using directional antennas to physically filter out the interference rather than using coding techniques. In other words, in JIM-Beam, the *antenna orientation, which is easy to conceal, is hidden from the adversary*, whereas USS techniques seek to *conceal the transmission frequency* from malicious jammers, even though the transmission frequency can be determined by the receiver in a timely manner. Thus, unlike previously proposed USS approaches, JIM-Beam does not need to constrain the computation capability of an attacker or its reaction time. Moreover,

JIM-Beam can be used in any narrow-band system, unlike spread spectrum (coordinated or otherwise) systems, which require a wide frequency spectrum in order to spread the transmitted signals. If a wide spectrum is allocated, JIM-Beam can be combined with spread spectrum techniques to offer even better jamming mitigation ability.

Like USS techniques, the goal of JIM-Beam is to mitigate the jamming attack and to establish communication channels without any pre-shared keys. The resulting communication channels can, in turn, be used to bootstrap other security properties, such as authentication (by setting up a key), or availability (by setting up a spreading code). However, optimizing the network capacity is specifically *not* the goal of JIM-Beam.

This chapter makes the following contributions:

1. I propose JIM-Beam, a jamming mitigation protocol that exploits spatial randomness. I also discuss several benefits of JIM-Beam over uncoordinated spread spectrum techniques. Additionally, as an orthogonal approach, JIM-Beam can combine uncoordinated spread spectrum techniques to provide better jamming mitigation.

2. I theoretically analyze the feasibility of JIM-Beam and perform Monte Carlo simulations of a flooding network to understand JIM-Beam's benefits when jammers are present.

3. I analyze the security of JIM-Beam and show that JIM-Beam can mitigate reactive jamming.

5.1 System and Attacker Models

In this section, I detail my system model and my attacker model. Specifically, I detail the hardware requirements of my protocol and discuss the attack my protocol seeks to mitigate.

5.1.1 System Model

In my protocol, a network device is equipped with at least one directional antenna and maybe one omni-directional antenna, depending on the variant

of my protocol in use. I analyze my protocol in the scenario where each device has only one directional antenna; however, my protocol can be readily extended to use multiple directional antennas. Moreover, I assume that the orientation of the main lobe of the directional antenna is steerable. I discuss antenna choices in detail in Sections 5.4.1 and 5.4.3.

As an alternative to a steerable antenna, a device can be equipped with multiple directional antennas that partition the space; that is, the main lobe of each antenna covers a disjoint portion of the space, and the antennas together cover the entire space. In this special case, the node can emulate an omni-directional antenna by using all sectors simultaneously. Steering the main lobe is then equivalent to turning on the appropriate antenna. This is a special case where neither an omni-directional antenna nor a steerable antenna is necessary.

I also assume that benign network devices are half-duplex. That is, whenever a benign device is transmitting, it cannot receive due to self-interference. Choi et al. recently propose that single-channel radios can be made full-duplex [68]. While full-duplex radios are able to improve the throughput of the network, the use of such radios does not fundamentally alter the operation of my protocol. In particular, while full-duplex enables the possibility of reactive jamming, it does not have a significant impact on a node's ability to avoid being jammed.

5.1.2 Attacker Model

I consider an attacker that is interested in denying availability to the network. The jammer reaction and computation assumptions are very strong. Other than the requirement of *causality*, in which an attacker cannot react to a future event, I do not limit the *computation ability* of an attacker or place any restriction on the amount of time between when an attacker hears a message and when it is able to start jamming in response to that message. Higher-layer protocols may use conventional cryptographic mechanisms, such as encryption or digital signatures, but JIM-Beam itself does not rely on any form of computational security.

I consider an attacker that injects interference into the communication channel. The injected interference can be either noise-like (i.e. jamming) or

correctly modulated with incorrect data (i.e. corruption by overwriting). I do not distinguish between these two cases since if either form of interference shadows the original data packet, the overlapping portion of the original data packet would be corrupted with high probability. I also do not specifically consider any replay attacks. An attacker that replays packets helps with delivering those packets, and is not in the attacker's interest.

5.2 Related Work

Prior studies propose using directional antennas to avoid interference and collisions in order to improve network performance. Zander analyzes a *forward progress* metric and concludes that in a slotted-ALOHA multi-hop network, using directional antennas with optimal orientation control can improve its performance greatly by reducing the risk of destructive packet collisions [69]. Yi et al. analyze the capacity of a network using a directional antenna for sending, receiving, or both [70]. The authors concluded that receiving using a directional antenna improves the network capacity by avoiding interference, and that transmitting using a directional antenna further improves the network capacity by providing better spatial reuse. These theoretical studies motivate the feasibility of JIM-Beam's approach; however, JIM-Beam's main goal is to provide an available channel in networks without any pre-shared keys and does not focus on network capacity.

Lu et al. [71] propose that transmitters should use directional antennas in order to conceal their locations, since attackers outside the main-lobe are less likely to detect the presence of the transmitter. The goal of my protocol is not to conceal the transmitter's presence, but to mitigate jamming. Thus, I instead propose that the receivers use directional antennas to receive packets.

Nikolaidis et al. [72] and Sani et al. [73] propose adaptively changing the directional antenna orientation in order to minimize the received interference. While these protocols work well against a slow-moving source of interference, a sophisticated set of jammers may be able to coordinate and force the nodes in the network to orient their directional antennas in a particular direction and subsequently disrupt the network availability. My approach thus suggests using randomized orientation so as to increase the required resources for jamming to be successful.

Noubir proposes using a sectored directional antenna to maintain network connectivity under jamming [74]. This prior study focuses on the optimal placement of jammers and the minimum number of jammers required to reduce the connectivity index of the network to 0. The author empirically concluded that in order to reduce the network connectivity index to 0, the number of jammers appears linearly proportional to the number of antenna sectors. My work, on the other hand, focuses on a flooding algorithm that provides physical-layer availability even under the jamming attack by taking advantage of the improved connectivity when network devices are equipped with directional antennas.

Many studies also employ spread spectrum techniques in order to avoid interference [58, 59, 60]. However, two devices can use spread spectrum to communicate only if the receiver can decode the transmission. Traditionally, the transmitter and the receiver encodes and decodes, respectively, using a shared key [58].

Recently, Strasser et al. propose the uncoordinated frequency hopping (UFH) scheme, which randomizes the frequency channels occupied in a slow-frequency-hopping spread spectrum scheme [62]. In UFH, the transmitter randomly selects a frequency channel on which to transmit, and each receiver also randomly selects a frequency channel on which to listen. If the transmitter and the receiver choose the same frequency channel at the same time, an event called *rendezvous*, the receiver is able to receive the transmission given sufficiently high SINR. Many follow-up studies extend the randomization idea to other coding techniques [59, 63, 64]. The common theme of this line of research is to use randomization in coding so that by the time the jammer learns the spreading code, the receivers have already received (although not necessarily have decoded) the associated packet; thus a jammer cannot reactively jam the signal. In other words, the goal of these schemes is to *conceal the code space on which a transmitter operates*, so that the jammer cannot discover that code and jam the message until after the message has been successfully received. My protocol instead randomizes the orientation of the main lobe of the directional antenna to *conceal the direction in which a receiver is listening*, so that even if the jammer can hear the transmission, the receivers have a chance of eliminating the malicious interference. My strategy of hiding the reception allows my protocol to be able to defend against

reactive jamming without making any assumptions about the computation ability or reaction time of a jammer.

5.3 JIM-Beam Flooding Protocol

I first present the basic JIM-Beam protocol, which introduces the basic concept of JIM-Beam without introducing any complicating optimizations. I then optimize JIM-Beam by adopting retransmissions and random backoff to provide persistent service that alleviates the broadcast storm; I also use directional antennas for both transmission and reception in order to take advantage of the better spatial reuse offered by directional antennas.

5.3.1 Basic JIM-Beam

The basic JIM-Beam scheme is similar to the basic flooding protocol: If a node receives a packet it has not seen before, the node rebroadcasts that packet so other nodes can also receive and pass it on.

In basic JIM-Beam, time is divided into time slots. At the beginning of a *frame*, consisting of T time slots, each network node randomly selects a new orientation and re-orients the main lobe of its directional antenna. The node then opportunistically receives packets from its neighbors that are located in the randomly chosen direction. If a node receives a packet it has not seen before, the node immediately broadcasts that packet using its omni-directional antenna in the next time slot.

5.3.2 B(ackoff) R(epeat) T(ransmit)-JIM-Beam

In the basic JIM-Beam protocol, the nodes that have heard a data transmission are likely to be close to each other since they share a common neighbor (i.e. the transmitter). When these nodes immediately rebroadcast that message, they will cause a *broadcast storm*, which results in a large number of collisions. I thus incorporate into the JIM-Beam protocol the random-backoff technique that is known to effectively mitigate the broadcast storm. In BRT-JIM-Beam, before each transmission, a transmitter waits for a random num-

ber of time slots. Two nearby transmitters are thus likely to transmit at different times, reducing the number of collisions.

Since the main lobe is oriented randomly, a JIM-Beam receiver's directional antenna indiscriminatingly suppresses transmissions from both benign transmitters and malicious jammers. In other words, there is a non-zero chance that a receiving node cannot hear its transmitting neighbor because of suboptimal antenna orientation. Thus in BRT-JIM-Beam, the transmitter retransmits each packet several times. Probabilistic optimal antenna orientation enables a JIM-Beam receiver to receive from the transmitter similarly to how rendezvous enables UFH receivers to receive from the transmitter.

The use of retransmission can be replaced by *network coding*, where multiple packets that are awaiting transmission are combined together using techniques such as Raptor Codes. Using network coding to distribute data when links are temporally inconsistent is also proposed by Papadimitratos et al. in a vehicular ad hoc network setting to distribute the certificate revocation list [75].

Additionally, BRT-JIM-Beam allows a node to use the same directional antenna for both reception and transmission, thereby reducing the number of antennas needed for the protocol. In BRT-JIM-Beam, the transmitter also selects a random direction to orient the main lobe of its directional antenna for every frame.

In summary, the BRT-JIM-Beam protocol retains the same packet reception behavior as the basic JIM-Beam protocol. However, when a node wants to transmit a packet, either to initiate a new flood or to forward a packet that the node has not previously forwarded, the sender does not transmit the packet right away; rather, it performs backoff and repetition coding. Before each broadcast attempt, the node randomly chooses a backoff counter BC, $\mathrm{BC} \leftarrow_R [0, \mathrm{BC}_{\max}]$ (where the maximum backoff counter, $\mathrm{BC}_{\max}$, is a system parameter), and transmits using the directional antenna after BC time slots. The sender broadcasts each message B times, possibly coded with other packets, to give neighboring receivers a better chance of receiving the packet.

5.4 Discussion on JIM-Beam Parameters

The most important parameters of the JIM-Beam protocol are the beam-solid-angle, maximum directivity, and the lobe steering ability of the directional antenna. Other parameters that also impact the performance of the JIM-Beam protocol include frame duration and, in the BRT-JIM-Beam variant, the maximum backoff counter and number of repeated broadcasts for each packet. In this section, I discuss how each parameter impacts the JIM-Beam protocol and present a gain analysis and a connectivity analysis.

5.4.1 Beam-Solid-Angle and Maximum Directivity of an Antenna

The beam-solid-angle and the maximum directivity are two important factors that determine the antenna's ability to mitigate interference. Given an antenna radiation pattern, the beam-solid-angle Ω_A is defined to be the integral of the normalized radiation intensity over the entire space. Equivalently, if an antenna concentrates all its radiation into a solid cone, the cone would have a solid angle of Ω_A. Considering only a 2-dimensional plane, an omni-directional antenna has beam-solid-angle of 2π. The beam-solid-angle is inversely proportional to the probability that a node receives from a randomly placed signal/interference source.

The directivity of an antenna in a certain orientation is defined to be the antenna's radiation in that orientation divided by the average radiation of the antenna. Thus, the maximum directivity of an antenna is equal to the peak-to-mean ratio of the antenna's radiation pattern. When a benign transmitter is optimally located within the main lobe and a single interference source is outside the main lobe, then on average the benign transmission enjoys a gain equaling the antenna's maximum directivity. In other words, the antenna directivity is similar to the coding gain in spread spectrum systems: correct orientation (code choice) provides antenna (processing) gain.

To illustrate the definition of beam-solid-angle and directivity, I consider two commercial directional antennas: an Anritsu 2000-1414 antenna[1] and an SMC DIFP18 antenna.[2] The radiation patterns included in the data sheets

[1] data sheet: http://www.anritsu.co.uk/files/11410-00376.pdf
[2] data sheet: http://www.smc.com/files/AP/DS_ANT.pdf

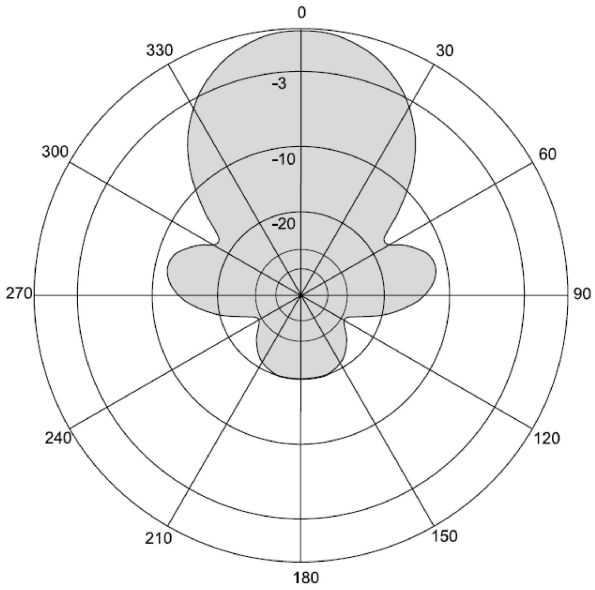
are shown in Figure 5.1. The Anritsu 2000-1414 antenna has a beam-solid-angle $\Omega_A = 0.163 \cdot (2\pi)$ and a maximum directivity of 6.138. The SMC DIFP18 antenna has a beam-solid-angle $\Omega_A = 0.042 \cdot (2\pi)$ and a maximum directivity of 23.668.

5.4.2 Gain Analysis

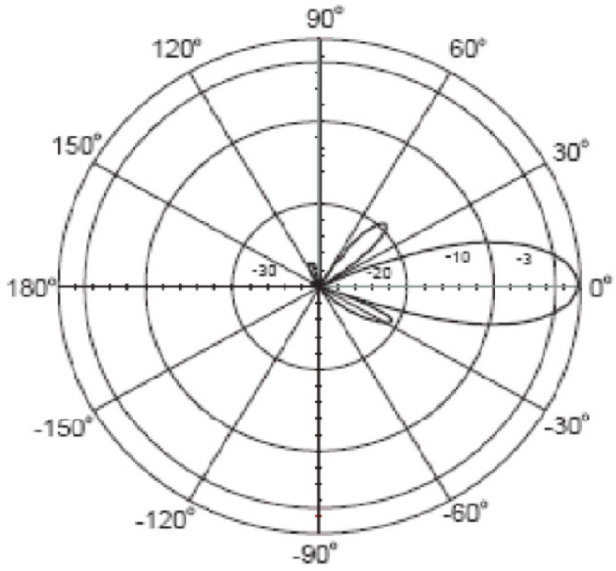
In this section, I study the ratio between the received power of the intended signal and the received power of the interference. In particular, I compare the antenna gain from using a directional antenna with the processing gain from using frequency hopping (FH) and direct sequence spread spectrum (DSSS).

I consider a case where a receiver using an omni-directional antenna and no coding would receive equal power from the intended transmitter and from a malicious jammer. The jammer is aware of the modulation scheme used by the transmitter. For each interference mitigation technique, I consider a different attack strategy. For DSSS, the jammer emits a uniformly-random level of voltage for each chip. For FHSS, the jammer selects a number of hopping patterns on which to emit power uniformly at random and divides his power equally across those random channels. When two of the attacker's hopping patterns use the same frequency channel for a single chip, the jammer doubles his transmission power in that channel. Against directional antennas, a randomly located jammer constantly emits power into the channel. When the main lobe of the receiver's directional antenna is oriented away from the jammer, the jammer's interference is reduced.

Since the antenna gain is related to the directivity of a directional antenna, I choose the chip length of a spread spectrum code to be the maximum directivity, rounded up to the next integer. Again, I consider the two directional antennas mentioned earlier: the Anritsu 2000-1414 antenna and the SMC DIFP18 antenna, with directivity of 6.14 and 23.67, respectively. I then perform a Monte Carlo analysis to compare the antenna gain of the Anritsu directional antenna to the processing gain of 7-chip FHSS and DSSS systems. I similarly compare the SMC antenna to 24-chip FHSS and DSSS systems. To show the best-case scenario for a directional antenna system, I also consider in each case the performance of a solid-beam sectored antenna that has equal directivity but no side lobe; that is, it has zero gain outside of its solid



(a) Radiation pattern of Anritsu 2000-1414



(b) Radiation pattern of SMCANT-DIFP18

Figure 5.1: Radiation patterns of two commercial directional antennas

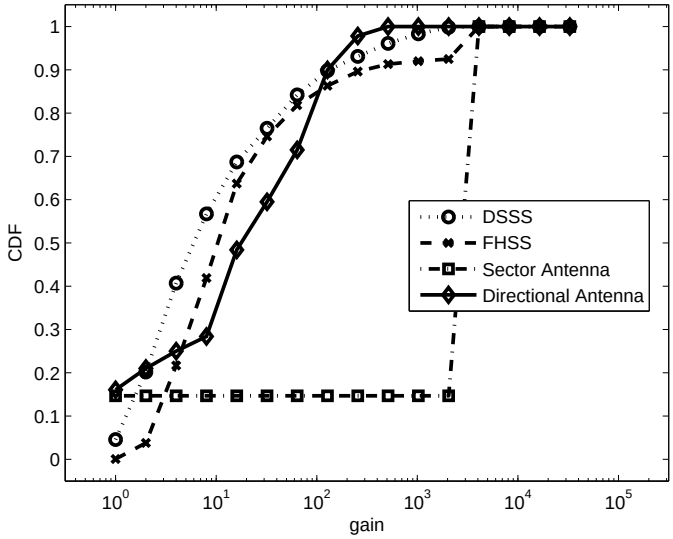
beam. In each case, I include a minimum noise floor at a level equal to the minimum gain of each commercial antenna; in the Anritsu case this level is 28 dB below the transmission power, and in the SMC case this level is 30 dB below the transmission power.

Figure 5.2 shows the cumulative distribution function of my results. The x-axis of each plot shows the gain (antenna or processing) enjoyed by each jamming mitigation system on a logarithmic scale, and the y-axis shows the probability that the gain will be less than the x-axis. Thus curves that are farther to the right reflect better performance. The figure shows that directional antennas occasionally offer no gain at all since the probability that the attacker is located within the orientation of the main lobe is higher than the probability that the attacker's random guesses are highly correlated with the transmitted spreading code. Furthermore, the maximum gain achievable by a commercial antenna is less than the maximum gain in spread spectrum systems because the commercial antenna's interference rejection ability is limited by its minimum gain in the side lobe. However, at the median and for over 60% of the time, the directional antenna is clearly better than using either spread spectrum technique, as shown by the CDF being farther to the right at $y \in [0.25, 0.85]$ when using the Anritsu antenna. Furthermore, with increased directivity, the directional antenna scheme provides better results.

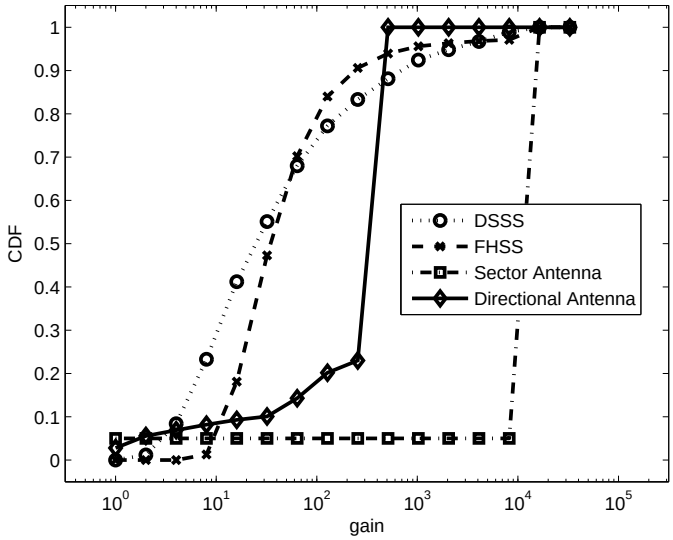
5.4.3 Steering the Main Lobe of a Directional Antenna

Because JIM-Beam uses randomized directional antenna orientation for interference rejection, each JIM-Beam node must be able to steer the main lobe of its antenna so that the main lobe can be randomly oriented over time. There are two means by which a node can steer its main lobe: The node can physically (mechanically) reorient its directional antenna so that the main lobe is also reoriented, or the node can electrically change the property of the antenna so that its main lobe is reoriented without physically moving the antenna.

The two commercial antennas I discussed in Section 5.4.1, Anritsu 2000-1414 and SMC DIFP18, do not have reconfigurable main lobes. Thus, to reorient these antennas, the nodes can rotate them by mounting them on a motor. Although motorizing a directional antenna can provide the reorienta-



(a) CDF of antenna and processing gains. Antenna directivity = 6.14, spreading code length = 7 chips.



(b) CDF of antenna and processing gains. Antenna directivity = 23.67, spreading code length = 24 chips.

Figure 5.2: Comparison of the CDF of the antenna and processing gain obtained from antenna directivity and spreading.

tion required by JIM-Beam, it has several drawbacks. A motorized antenna incurs additional energy consumption, which may be undesirable in networks sensitive to energy consumption, such as sensor networks. Motors can also generate emissions in fume and heat, which may present a side channel that an attacker can exploit.

Certain directional antennas can be reconfigured electrically. For example, an antenna array is steerable by exciting each individual element carefully. Boerman and Bernhard present a *pattern reconfigurable antenna* that is made of two parasitic strips with reconfigurable lengths. The authors show that the antenna has a 60° beam-width and the orientation of the main lobe can change by ±35° [76]. As the field of reconfigurable antenna matures, I believe such designs can greatly improve the performance and security of the JIM-Beam protocol.

5.4.4 Frame Duration

Depending on the extent to which nodes are time-synchronized, JIM-Beam offers great freedom in partitioning time into time slots and frames. If all network nodes are weakly time-synchronized, then all nodes can agree on a schedule in which time is divided into time slots. In each time slot, there is a transmission portion and a setup portion. In the transmission portion of a time slot, each node transmits or listens according to the JIM-Beam protocol, and in the setup portion of a time slot, each node reorients its directional antenna. The frame duration in this synchronized case is one time slot.

On the other hand, if network nodes are not time-synchronized, then the receiver should stay on a single orientation for at least the duration of two data packet transmissions. If the receiver stayed on an orientation only for one time slot, then the sender and receiver may not have synchronized packet start times, so the receiver would move away before it finishes its packet reception. To reduce the chance that a receiver will reorient away from a sender during a packet transmission, the frame duration should be two or more time slots. The period of time that a receiver can safely stay on a single orientation depends on the ability of the jammer to find that orientation.

5.4.5 Maximum Backoff Counter

To avoid the broadcast storm problem, BRT-JIM-Beam uses the backoff mechanism to spread out transmissions in time. The optimal choice of maximum backoff counter BC_{max} depends on the network density and any latency requirement. In a dense network, since more nodes may be waiting to transmit, the maximum allowed backoff counter should be greater than that used in a sparse network in order to avoid heavy collisions. However, excessive backoff impacts the performance of the network due to increased transmission latency for each packet. Thus, the optimal BC_{max} is the smallest one that satisfies the network quality-of-service requirements.

5.4.6 Number of Repeated Broadcasts for Each Packet

Since each JIM-Beam node randomly selects the orientation of its directional antenna, a receiver may be located within the transmission range of a neighboring node, and yet still not be able to receive the packet. BRT-JIM-Beam thus uses rebroadcasts to reduce the probability that a direct neighbor does not receive the packet. Although rebroadcasting persistently diminishes the probability that a neighboring node misses a data packet due to suboptimal orientation, it also increases the probability of collisions in a dense network. As a general rule, to avoid incurring collisions, the optimal number of rebroadcasts is negatively correlated to both the beam-solid-angle of the directional antenna and the network density.

5.4.7 Connectivity Analysis

In this section, I analyze the connectivity of a JIM-Beam node. In particular, I analyze the probability that a JIM-Beam receiver receives a single message without collision by considering the number of benign transmitters, the number of jammers, the number of repeated broadcasts, and the maximum backoff counter.

For my analysis, I consider a receiver equipped with a sectored directional antenna and located at the center of a circular field. The receiving node experiences a certain noise floor, which limits its receiving range. I then consider the free-space model with a *decoding threshold* of $\delta = 2 = 3$ dB;

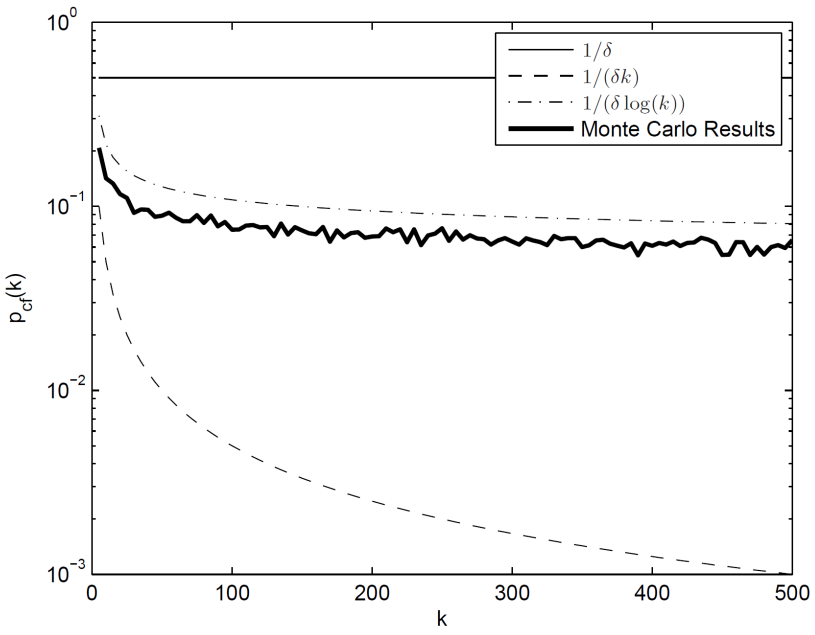


Figure 5.3: Monte Carlo simulation result of the collision-free probability

that is, if the power of the strongest signal received is 3 dB stronger than the sum of power from all other transmissions, the strongest signal is declared received. When only two packets are sent simultaneously, they *collide* if and only if both packets are received at the same time with signal strengths within 3 dB of each other. I simulate $N+J$ homogeneous transmitting nodes within the circular field, out of which N are benign and J are malicious jammers. The receiver can receive a message only if a transmission *from a benign transmitter* is more powerful than the sum of all other transmissions by the decoding threshold.

I first ignore the impact of using a sectored antenna, and try to understand the behavior of the collision probability with respect to the number of nodes. In Section 5.4.8, I derive two loose bounds on the probability that a node receives one transmission; the collision-free probability $p_{cf}(k)$ is bounded by $\frac{1}{k\delta} \leq p_{cf}(k) \leq \frac{1}{\delta}$, where k is the number of transmitters in the field and δ is the signal-to-noise threshold for decoding. Figure 5.3 shows these two loose bounds, $\frac{1}{\delta \log(k)}$, and the empirical results from my Monte Carlo simulation.

I now take into account the impact of using a directional antenna. Let $n < N$ be the number of benign nodes in the receiver's main lobe and $j < J$ be the number of jammers in the receiver's main lobe. I also model the backoff mechanism with maximum backoff counter $\mathrm{BC}_{\max}$. Given that the sectored

antenna has a beam-width of Ω, the normalized beam-width is defined to be $\Delta = \frac{\Omega}{2\pi}$, which is the fraction of space covered by the beam width. I also define $b(K, k, q)$ as the probability that, when each of K elements is chosen with probability q, exactly k elements are chosen. Specifically, b follows the binomial distribution:

$$b(K, k, q) = \binom{K}{k} q^k (1-q)^{K-k}.$$

Then the probability that a node receives a packet without collision is given by

$$\sum_{n=0}^{N} \sum_{j=0}^{J} \sum_{t=0}^{n} f(n, t, j) \left(\frac{t}{t+j} \right) p_{cf}(t+j),$$

where $f(n, t, j) = b(N, n, \Delta) b(J, j, \Delta) b\left(n, t, \frac{1}{\mathrm{BC}_{\max}}\right)$.

The $f(n, t, j)$ term signifies that out of N benign transmitters and J jammers, n and j are within the antenna beam, respectively, and out of n benign transmitters, t are transmitting and $n - t$ are still waiting for the backoff process. Since there are $t + j$ transmitters within the reception range, the collision-free probability is $p_{cf}(t + j)$; additionally, only benign transmitters send useful packets, giving rise to the $\frac{t}{t+j}$ term.

The probability of receiving at least one out of B repeated broadcasts of a single packet can then be approximated as:

$$\mathrm{P}[\text{receive}] = 1 - (1 - \mathrm{P}[\text{no collision}])^B.$$

5.4.8 Bounds on $P_{cf}(k)$

In this section, I derive the upper and lower bounds on $P_{cf}(k+1)$ with respect to k. We can make two key observations: First, if the i^{th}-closest transmitter is at distance d_i, then if $d_2 \leq d_1\sqrt{\delta}$, then the ratio between the closer transmitter's power and the farther transmitter's power will be less than δ (assuming free-space propagation loss, which is proportional to d^2), which ensures the loss of both packets. Second, if $d_i \geq d_1\sqrt{k\delta}$ for all transmitters other than the closest, then a collision is impossible, because each of k transmitters contributes power less than $\frac{1}{k\delta}$ of the closest transmitter. The sum of all interfering traffic, then, is less than $\frac{1}{\delta}$ of the power of the closest

transmitter, so the packet is guaranteed to be received (without considering the effect of noise).

The *reception range* of a receiver is defined as the distance from the transmitter at which the received signal power is δ times higher than the experienced noise. I denote the reception range of the receiver as r, and let the field be a circle with radius $R \geq r$.

First note that the transmitters are scattered uniformly and independently across the field; additionally, the events "transmitter t_i is closest to the receiver" and "transmitter t_j is closest to the receiver" are disjoint for $j \neq i$ (i.e. they cannot both happen at the same time). Thus, the collision-free probability is less than the probability that the second closest transmitter is $\sqrt{\delta}$ farther away from the receiver than the closest transmitter:

$$\begin{aligned} & p_{cf}(k+1) \\ \leq\ & (k+1) \int_{d_1=0}^{r} \frac{2}{R^2} d_1 \cdot \\ & \int_{d_2,d_3,\ldots,d_{k+1}=\sqrt{\delta}d_1}^{R} \left(\frac{2}{R^2}\right)^k \prod_{t=2}^{k} d_t \cdot dd_t \cdots dd_1 \\ =\ & (k+1) \int_{d_1=0}^{r} \frac{2}{R^2} d_1 \left(1 - \frac{\delta d_1^2}{R^2}\right)^k dd_1. \end{aligned}$$

By substituting $\gamma = 1 - \frac{\delta d_1^2}{R^2}$ and finishing the integration, obtain

$$p_{cf}(k+1) \leq \frac{1}{\delta}\left(1 - \left(1 - \frac{\delta r^2}{R^2}\right)^k\right).$$

Similarly, by using the second observation, obtain a lower bound:

$$p_{cf}(k+1) \geq \frac{1}{k\delta}\left(1 - \left(1 - \frac{\delta r^2}{R^2}\right)^k\right).$$

In the special case where $r = \frac{R}{\sqrt{\delta}}$, the upper bound reduces to $p_{cf}(k) \leq \frac{1}{\delta}$ and the lower bound reduces to $p_{cf}(k) \geq \frac{1}{(k-1)\delta}$. Empirically, in Section 5.4.7 I show the behavior of $p_{cf}(k)$ appears to be correlated with $\log(k)$.

5.5 Security Analysis of JIM-Beam

I analyze the security of JIM-Beam under two particular attacks:

1. Jamming and corrupting a portion of a data packet; and
2. Inserting false packets to exhaust computation.

For ease of analysis and only for this section, I analyze the security of the JIM-Beam protocol using a *sectored directional antenna*.

5.5.1 Jamming by Corrupting Part of a Packet

A data packet usually consists of some number of bits of actual data and is accompanied by some number of error correcting bits. The error correcting portion is usually much smaller in length than the actual data portion. The error correcting bits are used so that if a small number of bits are not successfully received because of erasures or noise, those bits can still be recovered. For example, a common Reed-Solomon code encodes 223 bytes of data with 32 bytes of error correcting code, so that one 255-byte packet can correct up to 16 bytes = 128 bits of errors.

A jammer does not need to interfere with the entire packet in order to disrupt connectivity, but only needs to corrupt a significant portion of the packet so that the packet is not recoverable by error correction coding. In the above Reed-Solomon example, a jammer only needs to corrupt 16 bytes, or 6.3%, of the packet in order to make it uncorrectable.

I analyze two scenarios. In the first scenario, the normalized beam-width, $\frac{\Omega}{2\pi}$, of the receiver's sectored directional antenna is *larger than* the error correction portion of the data packet (0.063 in the above Reed-Solomon example). In this case, a jamming strategy is to revolve around the receiver. If a jammer can make one complete revolution around the receiver over the duration of a data packet, the jammer can always corrupt a portion of the data packet large enough so as to render the packet uncorrectable. If a jammer can only make $\frac{\theta_j}{2\pi} < 1$ revolution over the duration of the packet, the jammer can corrupt a packet with probability $\frac{\theta_j+\Omega}{2\pi} < p < \frac{\theta_j+2\Omega}{2\pi}$. The duration of a packet transmission can be determined from the packet length and transmission rate.

I consider an example where a spherical node with radius 10 cm uses a sectored directional antenna with beam-width $\Omega = 60$ degrees to receive a data packet 1472 bytes in size transmitted at a rate of 11 Mbps. The packet size and transmission rate corresponds to common figures in a 802.11b network. The packet duration is $\frac{1472 \cdot 8}{11 \cdot 10^6} \approx 1.07$ ms. If a jammer wishes to revolve around the receiver in order to corrupt a data packet with probability 1, the jammer needs to make at least two-third of a revolution within 1.07 ms. Using Newtonian physics, this corresponds to a radial acceleration of $\alpha = \left(\frac{2/3}{0.00107}\right)^2 \cdot 0.1 = 3957$ g, where g is the earth's gravitational pull. As a point of comparison, a Formula One race car can turn[3] with a radial acceleration of around 6 g.

In the second scenario, the normalized beam-width of the receiver's sectored antenna is smaller than half of the error correction portion of the data packet. In this case, the ratio between the time a jammer must remain in a single antenna sector and the packet duration is larger than the ratio between the angular width of the sectored antenna and a full revolution. In other words, even if a single jammer has the ability to revolve around the network node once every packet duration (which requires extreme acceleration, as discussed above), the probability that he can corrupt a significant portion of the packet so as to make the packet uncorrectable is still less than one.

5.5.2 Packet Insertion Attacks

In JIM-Beam, as in previous randomized spread spectrum studies, a transmitting node and a receiving node communicate without a pre-shared key or schedule; and packets may be received out-of-order. A malicious attacker can thus seek to corrupt a data stream by inserting bogus packets. Exhaustively searching all packets for the correct stream is exponential in computation time and is infeasible for a receiver with realistic computational ability.

Because the problem of message injection is fundamentally a problem of message authentication, allowing packets to be sufficiently large so that an entire authenticator can fit inside the packet can mitigate the message injection attack. For example, current public key cryptography schemes such as RSA

[3] http://www.formula1.com/news/headlines/2009/9/10005.html

and ECDSA have signatures and certificates that are each no larger than 200 bytes; symmetric and unconditionally secure authentication (e.g. [77]) have authenticators that are typically on the order of 20 bytes. My scheme shows strong performance with 1472 byte messages at 11 Mbps, and can likely extend to larger packets (in my earlier example, 94 kB packets still require the jammer to revolve at over 6 g acceleration), which means that my scheme is compatible with *nearly every reasonable authentication scheme in the literature*. This is in contrast to USS protocols that require each frame to be shorter than the jammer's reaction time (e.g. [62]), in which case it may not be possible to insert a strong authenticator in each frame.

5.6 Evaluation

5.6.1 Methodology

We perform Monte Carlo simulations using MATLAB to study the effectiveness of JIM-Beam by measuring and comparing the packet delivery ratio between three different flooding protocols:

1. Flooding with backoff and retries, using omni-directional antennas for both transmitting and receiving;
2. BRT-JIM-Beam; and
3. Uncoordinated frequency hopping (UFH).

I perform two sets of simulations for BRT-JIM-Beam and UFH. In the first set, I equip each BRT-JIM-Beam node with an Anritsu 2000-1414 directional antenna, and I let each UFH node randomly choose to transmit or listen on one of seven orthogonal frequency channels. In the second set, I equip each JIM-Beam node with an SMCANT-DIFP18 directional antenna, and let each UFH node choose from 24 orthogonal frequency channels.

For all protocols, I assume that a node is able to receive a message if the power of that message is higher than the sum of interferences by the decode threshold $\delta = 2 = 3$ dB. In the BRT-JIM-Beam simulations, I take into account all interference sources including noise, fading, and signals from both the main-lobe and the side-lobes of the directional antenna.

I define distances in terms of units of transmission distance: When a single transmitter and single receiver are separated by one unit, and no other nodes are transmitting, the signal power at the receiver is 3 dB higher than the noise power (which consists only of the noise floor). Since my decode threshold is 3 dB, a unit represents the distance at which a packet can be received with 50% probability.[4] In my simulation, I have a field that is 4 unit-distance by 4 unit-distance. For all simulated protocols, I choose a maximum backoff counter $\mathrm{BC}_{\max} = 10$, and 5 rebroadcasts for every packet. I randomly select a benign node as the initiator, and let it transmit a data packet.

I assume a homogeneous network in which all nodes transmit at the same power level. I uniformly randomly distribute benign nodes and jammers on the field. The number of benign nodes ranges from 8 to 800 nodes, corresponding to a node density of 0.5 to 50 nodes per square unit-distance. In addition to the benign nodes, the number of jammers ranges from 0 to 50% of the number of benign nodes. For each combination of protocol, number of benign nodes, and number of attackers, I performed 250 runs of simulation.

I let each jammer's transmission power equal a benign transmitter's transmission power. A jammer that can emit a higher level of power can be considered as several weaker colluding jammers at the same location. I simulate the following jammer strategy:

1. The jammers remain stationary on the field;
2. The jammers jam using omni-directional antennas; and
3. In UFH simulation, each jammer chooses one frequency channel on which to inject interference.

Previous study discussed in detail the number of frequency channels on which the jammers should jam in order to degrade system performance the most [62]. Thus, my one-channel jamming strategy presents an upper bound on the performance of the UFH system.

All transmissions, benign or malicious, are subject to fading, and I adopt a Rician fading channel with parameter $K = 10$. In a Rician channel, a signal is split into two portions: a line-of-sight portion and a non-line-of-sight multi-path portion; the ratio in power between these two portions is

[4]Since noise from the noise floor is random, it can help or hurt a packet; thus packets can be lost inside of a unit transmission distance and can be received outside a unit transmission distance, with nonzero probability.

K. In other words, in my Rician fading channel, the line-of-sight signal is 10 times more powerful than the multi-path signal. Previous measurement work by Puccinelli and Haenggi suggest $K = 17.5$ in an indoor environment [78]; thus my assumption of $K = 10$ is a conservative choice. I also assume the signal suffers a path loss exponent of $\sigma = 2.2$. That is, doubling the distance between transmitter and the receiver makes the received transmission $(1 - 2^{-2.2}) = 78\%$ weaker (this figure would be 75% in the free-space model).

I calculate the packet delivery ratio by calculating the fraction of benign nodes that have received the data packet at the time the flooding stops. In particular, if all benign nodes received the data packet, the packet delivery ratio is 1; if no nodes other than the initiator received the data packet, and there are N benign nodes, then the packet delivery ratio is $1/N$.

5.6.2 Monte Carlo Simulation Results

Figure 5.4, Figure 5.5, and Figure 5.6 show the packet delivery ratio (PDR) versus the density of benign nodes on the field. The error bars represent the 95% confidence intervals across my 250 runs per data point. I plot a different curve for each different ratio between jammers and benign nodes. Observe that with all protocols, intuitively, the packet delivery ratio decreases with respect to the number of jammers.

Figure 5.4 shows the performance of a flooding network using random backoff and multiple retries for each forwarding message, but without any mechanisms to mitigate jamming. Observe that even a small number of jammers is able to disrupt the connectivity of the network. Specifically, even at $j = 10\%$, the PDR drops to less than 0.4.

Figure 5.5 and Figure 5.6 show the performance of the BRT-JIM-Beam and the UFH protocols. From these figures, one can make the following observations:

1. **In sparse networks, BRT-JIM-Beam significantly outperforms UFH**. From the left side of Figure 5.5(b), observe that when nodes are sparsely populated, the UFH protocol hurts the PDR because the few neighboring nodes must be both within range and listening to the correct frequency channel. However, using a directional antenna for transmission extends the range of the transmitter; thus, BRT-JIM-Beam is

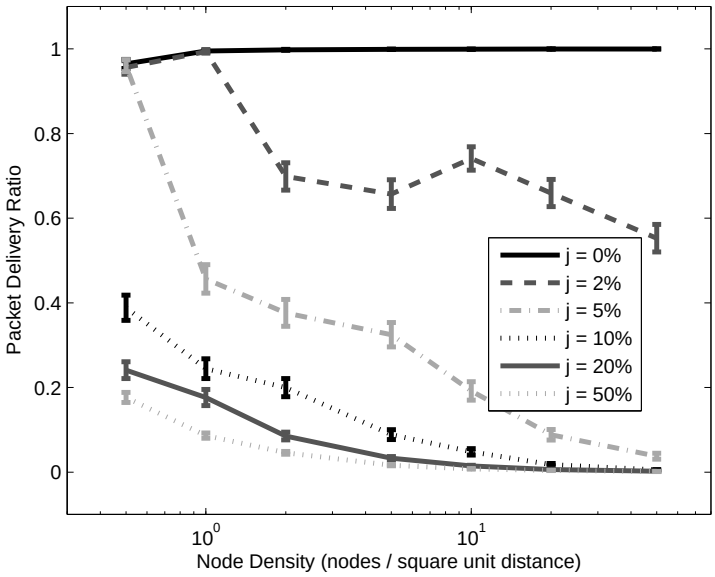
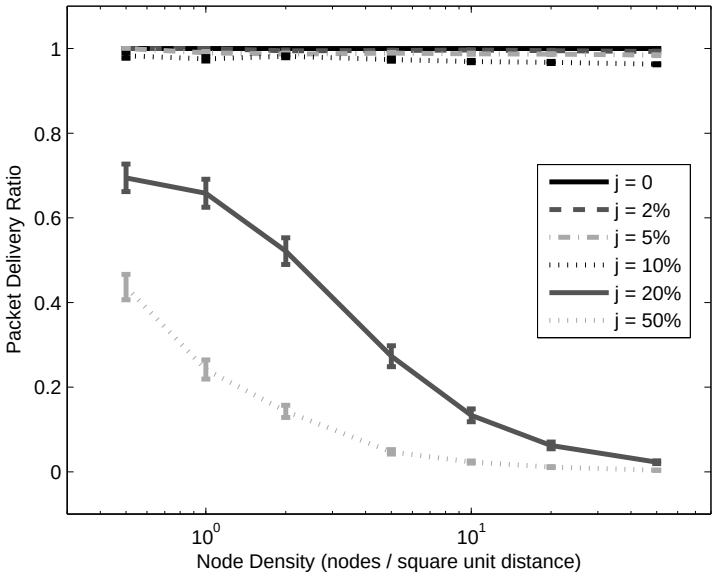


Figure 5.4: Packet delivery ratio of flooding with backoff and retries

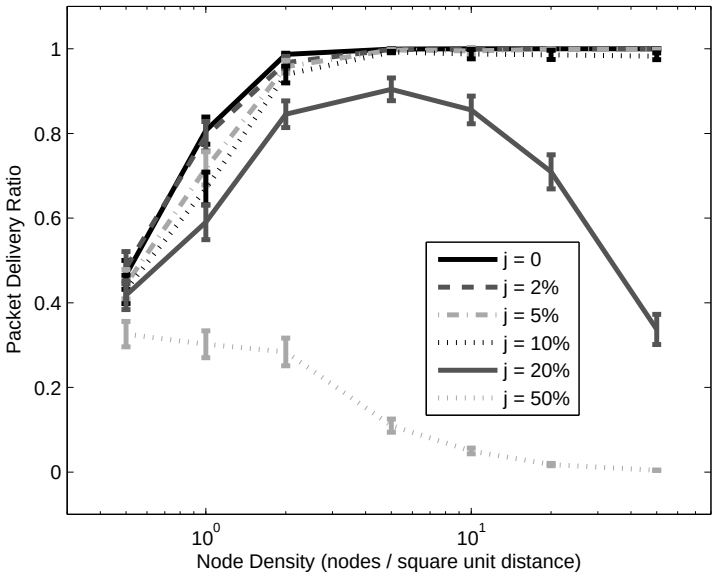
able to reach more nodes even in sparse networks as observed from Figure 5.5(a). This observation is further supported by Figure 5.6(a) and Figure 5.6(b).

2. **In overly dense networks, BRT-JIM-Beam and UFH both suffer from broadcast storm.** Since neither BRT-JIM-Beam nor UFH employ any handshake mechanism, transmitters that are within range of each other would inevitably interfere with each other occasionally. Thus, in an overly dense network where a large number of transmitters are clustered together, the performance of both schemes deteriorates.

3. **BRT-JIM-Beam is able to use less power than UFH to provide comparable jamming resilience.** Since I define distance based on a node's transmission range, a dense network can transform itself to a sparse network by reducing the transmission power of the nodes. Hence, to achieve the same level of PDR, the required transmission power of each BRT-JIM-Beam node is lower than that of a UFH network. This property is highly desirable to networks that are sensitive to energy consumption.

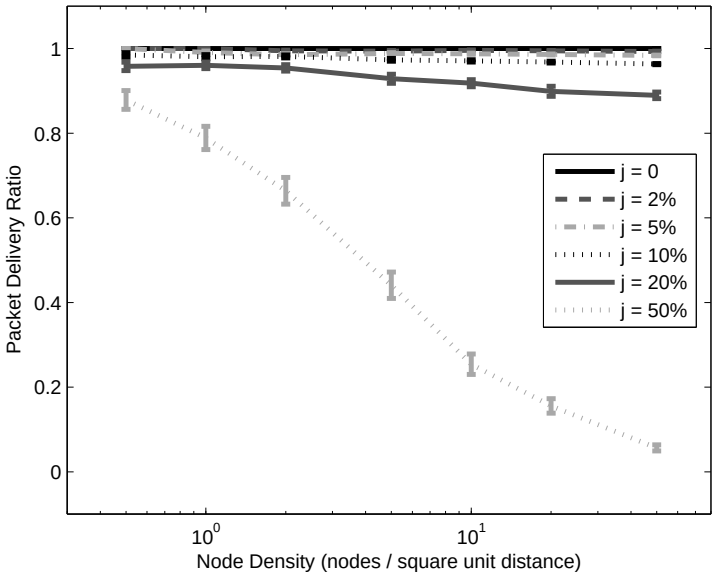


(a) Flooding using BRT-JIM-Beam with Anritsu 2000-1414

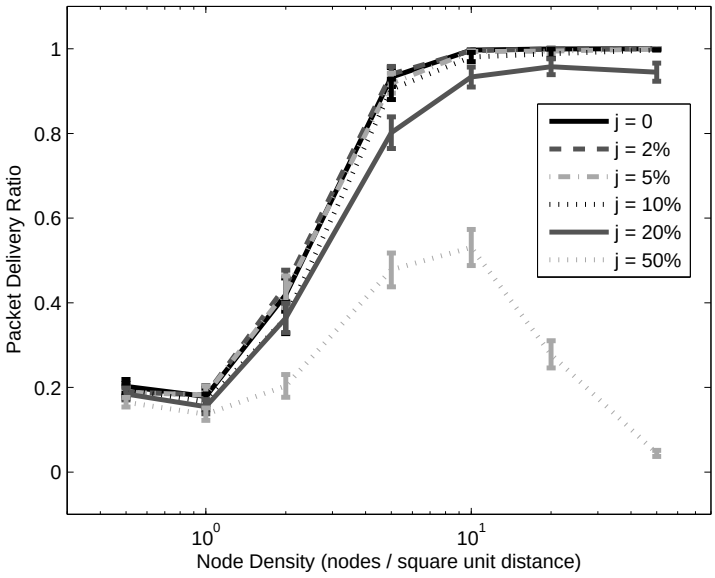


(b) Flooding using UFH with seven frequency channels

Figure 5.5: Comparative results of the packet delivery ratio between JIM-Beam and UFH ($\Delta = 6.138, \eta = 7$)



(a) Flooding using BRT-JIM-Beam with SMCANT-DIFP18



(b) Flooding using UFH with 24 frequency channels

Figure 5.6: Comparative results of the packet delivery ratio between JIM-Beam and UFH ($\Delta = 23.67, \eta = 24$)

5.7 Integrating Other Anti-Jamming Schemes

JIM-Beam provides strong jamming mitigation capabilities whenever the directional antenna is pointed away from the jammer. In particular, commercial off-the-shelf directional antennas allow my scheme to attenuate jammer signals by up to 28–30 dB. However, in certain environments, the jammer may have access to much more power than ordinary transmitters. In such environments, JIM-Beam can be combined with an existing anti-jamming mechanism. In this section, I explore how JIM-Beam's integration with other schemes can help improve the security of the overall system against very strong attackers.

When used in concert with another anti-jamming system, JIM-Beam limits the number of attackers that can have impact at any given time. This combination provides benefits to both JIM-Beam's performance and the performance of the anti-jamming system. In this section, I will use as a running example a Direct-Sequence Spread Spectrum code that sends 10 chips per bit.

When the jammer is out of the main lobe in which JIM-Beam is receiving, the attacker is attenuated by up to 30 dB. This increased attenuation means that the overlying anti-jamming system does not need to provide as much protection; for example, combining the 10 dB DSSS scheme with a 30 dB attenuation from JIM-Beam provides 40 dB of total jammer attenuation. To obtain 40 dB of gain from DSSS alone would require 10,000 chips per bit, or a factor of 1000 additional overhead. Likewise, to obtain 40 dB of gain from JIM-Beam alone may require an excessively directive antenna, which may also impact the probability of receiving from a benign transmitter.

When the jammer is in the main lobe in which JIM-Beam is receiving, DSSS is the sole source of jamming mitigation. However, because of directionality, DSSS needs to tolerate fewer jammers than if it were used alone. For example, if the directivity is 6.14, as in the Anritsu antenna, DSSS needs to handle only approximately one-sixth of the total number of jammers in expectation. Another way to say this is that if 17.8 dB is ordinarily needed to reject all jammers, then 10 dB together with an antenna with directivity of six is also sufficient, resulting in a six-fold decrease in DSSS overhead.

Finally, because JIM-Beam has minimal assumptions on attacker power, even if the overlying anti-jamming scheme is broken (for example, the spread-

ing code in use by DSSS is leaked), JIM-Beam can still provide the same level of performance as when it was used without any overlying anti-jamming scheme. In other words, if DSSS provides 10 dB of gain, then the receiver can reject up to 40 dB of jammer input; but even if DSSS provides no gain, the receiver can still reject up to 30 dB of jammer input.

In summary, there are four major advantages in combining JIM-Beam with other overlying anti-jamming schemes: First, it provides a lower-overhead way to reject very strong jammers as compared to using either JIM-Beam or the overlying anti-jamming scheme alone; second, it allows the system to reject weak jammers that happen to be in the main lobe as long as the anti-jamming scheme provides sufficient gain; third, it reduces the number of jammers that the overlying anti-jamming scheme needs to reject; and finally, it provides a fail-safe source of jammer rejection in case the overlying anti-jamming scheme suffers a security failure.

5.8 Chapter Summary

In this chapter, I propose using random directional antenna orientation to mitigate the jamming attack. Previously proposed uncoordinated spread spectrum schemes use randomization to *conceal the code space on which a transmitter operates*, relying on the jammer's limited computational ability so that the jammer cannot discover that code and jam the message until after the message has been successfully received. Instead, I propose the JIM-Beam protocol that randomizes the orientation of the main lobe of a directional antenna to *conceal the direction in which a receiver is listening*. One particular benefit of using directional antennas to reject interference is that jammers cannot perform the equivalent of wideband jamming, because the location of each malicious node is a single-source of interference.

JIM-Beam is orthogonal to spread spectrum techniques, and when JIM-Beam is used in conjunction with an overlying anti-jamming scheme, the combined system has four major advantages:

1. It provides a lower-overhead way to reject very strong jammers as compared to using either JIM-Beam or the overlying anti-jamming scheme alone;

2. It allows the system to reject weak jammers that happen to be in the main lobe as long as the anti-jamming scheme provides sufficient gain;

3. It reduces the number of jammers that the overlying anti-jamming scheme needs to reject; and

4. It provides a fail-safe source of jammer rejection in case the overlying anti-jamming scheme suffers a security failure.

I use MATLAB simulation to evaluate the effectiveness of the JIM-Beam protocol, and show that BRT-JIM-Beam can significantly outperform the previously proposed UFH protocol in sparse networks. For networks that are sensitive to energy consumption, BRT-JIM-Beam is also able to use less power while providing jamming mitigation comparable to UFH.

CHAPTER 6

CONCLUSION

In this dissertation, I consider three philosophies in providing availability in the lower layers of a network. Each philosophy has its own advantages in comparison to others; and I present several case studies to illustrate these philosophies.

A real-life network can use a combination of techniques derived from these defense philosophies to achieve the desired reliability properties. For example, a network can use fairness enforcement as a fail-safe addition to a detection scheme; the network can also use randomization as a bootstrap mechanism in face of powerful attackers. Designing and combining these techniques provide a rich set of networking research topics and will surely benefit tomorrow's networks greatly.

REFERENCES

[1] J. T. Chiang, J. J. Haas, and Y.-C. Hu, "Secure and precise location verification using distance bounding and simultaneous multilateration," in *Proceedings of the Second ACM Conference on Wireless Network Security (WiSec '09)*, 2009, pp. 181–192.

[2] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites," in *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*, 2002, pp. 293–304.

[3] J. Choi, J. T. Chiang, D. Kim, and Y.-C. Hu, "Partial deafness: A novel denial-of-service attack in 802.11 networks," in *Security and Privacy in Communication Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 50. Berlin, Germany: Springer Berlin – Heidelberg, 2010, pp. 235–252.

[4] D. Kim, J. T. Chiang, Y.-C. Hu, A. Perrig, and P. R. Kumar, "CRAFT: A new secure congestion control architecture," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*, 2010, pp. 705–707.

[5] J. C. Navas and T. Imielinski, "GeoCast – geographic addressing and routing," in *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '97)*, 1997, pp. 66–76.

[6] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," UCLA Computer Science Department, Tech. Rep. UCLA-CSD TR-01-0023, 2001.

[7] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 48–57, Feb. 2001.

[8] S. Brands and D. Chaum, "Distance-bounding protocols," in *Advances in Cryptology (EUROCRYPT '93)*, vol. 765, 1994, pp. 344–359.

[9] S. Čapkun and J. P. Hubaux, "Secure positioning of wireless devices with application to sensor networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, vol. 3, Mar. 2005, pp. 1917–1928.

[10] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *Proceedings of the ACM Workshop on Wireless Security (WiSe 2003)*, Sep. 2003, pp. 1–10.

[11] N. Chandran, V. Goyal, R. Moriarty, and R. Ostrovsky, "Position based cryptography," in *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '09)*, 2009, pp. 391–407.

[12] Y. Desmedt, R. Safavi-Naini, H. Wang, C. Charnes, and J. Pieprzyk, "Broadcast anti-jamming systems," in *Proceedings of the IEEE International Conference on Networks (ICON '99)*, Sep. 1999, pp. 349–355.

[13] S. Čapkun and J. P. Hubaux, "Secure positioning in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 221–232, Feb. 2006.

[14] D. Singelée and B. Preneel, "Location verification using secure distance bounding protocols," in *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, 2005 (MASS 2005).*, Nov. 2005, pp. 840–846.

[15] S. Čapkun, K. Rasmussen, M. Srivastava, and M. Cagalj, "Securing localization with hidden and mobile base stations," *IEEE Transactions on Mobile Computing*, vol. 7, no. 4, pp. 470–483, 2008.

[16] L. Cai, K. Zeng, H. Chen, and P. Mohapatra, "Good neighbor: Ad hoc pairing of nearby wireless devices by multiple antennas," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS 2011)*, Feb. 2011.

[17] D. Vook, B. Hamilton, A. Fernandez, J. Burch, and V. Srikantam, "An update on nanosecond-level time-synchronization with IEEE-1588," in *Proceedings of 2005 Conference on IEEE-1588 Standard for Clock Synchronization*, Oct. 2005.

[18] K. Ishikawa and A. Mita, "Time synchronization of a wired sensor network for structural health monitoring," *Smart Materials and Structures*, vol. 17, no. 015016, 2008.

[19] K. Rasmussen and S. Čapkun, "Realization of RF distance bounding," in *Proceedings of the 19th USENIX Security Symposium*, Aug. 2010, pp. 389–402.

[20] W.-Y. Lee, "Preliminary investigation of mobile radio signal fading using directional antennas on the mobile unit," *IEEE Transactions on Vehicular Communications*, vol. 15, no. 2, pp. 8–15, Oct. 1966.

[21] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11, 2007.

[22] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, Apr. 2003, pp. 836–843.

[23] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ: Wiley-Interscience, 2006.

[24] A. Kamerman and L. Monteban, "Wavelan-ii: A high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal*, vol. 2, no. 3, pp. 118–133, 1997.

[25] J. C. Bicket, "Bit-rate selection in wireless networks," M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA, 2005.

[26] S. Gupta, V. Shankar, and S. Lalwani, "Reliable multicast MAC protocol for wireless LANs," in *Proceedings of the IEEE 2003 International Conference on Communications (ICC'03)*, May 2003, pp. 93–97.

[27] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Selected Areas in Cryptography*, vol. 2259, 2001, pp. 1–24.

[28] A. Stubblefield, J. Ioannidis, and A. D. Rubin, "A key recovery attack on the 802.11b wired equivalent privacy protocol (WEP)," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, pp. 319–332, May 2004.

[29] A. Bittau, M. Handley, and J. Lackey, "The final nail in WEP's coffin," in *Proceedings of the 27th IEEE Symposium on Security and Privacy*, May 2006, pp. 386–400.

[30] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions," in *Proceedings of the 12th USENIX Security Symposium*, Aug. 2003, pp. 15–27.

[31] F. Ferreri, M. Bernaschi, and L. Valcamonici, "Access points vulnerabilities to DoS attacks in 802.11 networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'04)*, Mar. 2004, pp. 634–638.

[32] G. Tan and J. Guttag, "Time-based fairness improves performance in multi-rate WLANs," in *Proceedings of the USENIX Annual Technical Conference*, Jun. 2004, pp. 269–282.

[33] R. J. Aumann and M. Maschler, "Game theoretic analysis of a bankruptcy problem from the Talmud," *Journal of Economic Theory*, vol. 36, no. 2, pp. 195–213, Aug. 1985.

[34] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2002*, Nov. 2002, pp. 71–82.

[35] A. Hussain, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," in *Proceedings of ACM SIGCOMM 2003*, Aug. 2003, pp. 99–110.

[36] M. Thottan and C. Ji, "Anomaly detection in IP networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2191–2204, Aug. 2003.

[37] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-based IP traceback," in *Proceedings of ACM SIGCOMM 2001*, Aug. 2001, pp. 3–14.

[38] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *Proceedings of ACM SIGCOMM 2000*, Aug. 2000, pp. 295–306.

[39] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *Computer Communications Review*, vol. 32, no. 3, pp. 62–73, Jul. 2002.

[40] J. Ioannidis and S. M. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Proceedings of Network and Distributed System Security Symposium (NDSS '02)*, 2002.

[41] K. Argyraki and D. R. Cheriton, "Active Internet traffic filtering: Real-time response to denial-of-service attacks," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATEC '05)*, 2005, pp. 135–148.

[42] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer DoS defense against multimillion-node botnets," in *Proceedings of ACM SIGCOMM 2008*, 2008, pp. 195–206.

[43] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet denial-of-service with capabilities," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 39–44, 2004.

[44] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks," in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, May 2004, pp. 130–143.

[45] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," *SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 241–252, 2005.

[46] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: Protecting connection setup from denial-of-capability attacks," *SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 289–300, 2007.

[47] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," in *Proceedings of ACM SIGCOMM '98*, 1998, pp. 118–130.

[48] H. T. Kung and S. Y. Wang, "TCP trunking: Design, implementation and performance," in *Proceedings of the Seventh Annual International Conference on Network Protocols (ICNP '99)*, 1999, pp. 222–234.

[49] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," IEFT RFC 2581, Apr. 1999.

[50] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, "On routing asymmetry in the Internet," in *Global Telecommunications Conference 2005 IEEE (GLOBECOM '05)*, vol. 2, Nov. 2005, pp. 904–909.

[51] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP congestion control with a misbehaving receiver," *SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 71–78, 1999.

[52] F. Zhao and S. F. Wu, "Analysis and improvement on IPSec anti-replay window protocol," in *Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN 2003)*, Oct. 2003, pp. 553–558.

[53] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the First ACM Conference on Computer and Communications Security (CCS '93)*, 1993, pp. 62–73.

[54] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," IETF RFC 2104, Feb. 1997.

[55] D. R. Stinson, *Cryptography: Theory and Practice*. Boca Raton, FL: Chapman & Hall/CRC, 2002.

[56] J. Nagle, "Congestion control in IP/TCP Internetworks," IETF RFC 896, Jan. 1984.

[57] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 67–82, 1997.

[58] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, "Theory of spread spectrum communications - A tutorial," *IEEE Transactions on Communications*, vol. 30, no. 5, pp. 855–884, 1982.

[59] C. Pöpper, M. Strasser, and S. Čapkun, "Anti-jamming broadcast communication using uncoordinated spread spectrum techniques," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 5, pp. 703–715, Jun. 2010.

[60] J. T. Chiang and Y.-C. Hu, "Cross-layer jamming detection and mitigation in wireless broadcast networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 286–298, Feb. 2011.

[61] W. Bahn, L. Baird, and M. Collins, "The use of concurrent codes in computer programming and digital signal processing education," *Journal of Computing in Small Colleges*, vol. 23, no. 1, pp. 174–180, 2007.

[62] M. Strasser, C. Pöpper, S. Čapkun, and M. Cagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy (SP 2008)*, May 2008, pp. 64–78.

[63] T. Jin, G. Noubir, and B. Thapa, "Zero pre-shared secret key establishment in the presence of jammers," in *Proceedings of the Tenth ACM International Symposium on Mobile ad hoc Networking and Computing (MobiHoc 2009)*, May 2009, pp. 219–228.

[64] Y. Liu, P. Ning, H. Dai, and A. Liu, "Randomized differential DSSS: Jamming-resistant wireless broadcast communication," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM '10)*, Mar. 2010, pp. 1–9.

[65] A. Viterbi, "Spread spectrum communications: Myths and realities," *Communications Magazine, IEEE*, vol. 40, no. 5, pp. 34–41, May 2002.

[66] P. J. Crepeau, "A connection between Rayleigh fading and worst-case partial band interference," in *Proceedings of IEEE Military Communications Conference (MILCOM '89)*, vol. 3, Oct. 1989, pp. 693–696.

[67] A. Ali, "Worst-case partial-band noise jamming of Rician fading channels," *IEEE Transactions on Communications*, vol. 44, no. 6, pp. 660–662, Jun. 1996.

[68] J. I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti, "Achieving single channel, full duplex wireless communication," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking (MobiCom '10)*, 2010, pp. 1–12.

[69] J. Zander, "Slotted ALOHA multihop packet radio networks with directional antennas," *Electronics Letters*, vol. 26, no. 25, pp. 2098–2100, Dec. 1990.

[70] S. Yi, Y. Pei, and S. Kalyanaraman, "On the capacity improvement of ad hoc wireless networks using directional antennas," in *Proceedings of the Fourth ACM International Symposium on Mobile ad hoc Networking & Computing (MobiHoc '03)*, 2003, pp. 108–116.

[71] X. Lu, F. Wicker, P. Lio, and D. Towsley, "Security estimation model with directional antennas," in *Proceedings of the Military Communications Conference, 2008 (MILCOM 2008)*, Nov. 2008, pp. 1–6.

[72] G. Nikolaidis, A. Zhushi, K. Jamieson, and B. Karp, "Cone of silence: Adaptively nulling interferers in wireless networks," *SIGCOMM Computer Communications Review*, vol. 40, pp. 433–434, Aug. 2010.

[73] A. A. Sani, L. Zhong, and A. Sabharwal, "Directional antenna diversity for mobile devices: Characterizations and solutions," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking (MobiCom '10)*, 2010, pp. 221–232.

[74] G. Noubir, "On connectivity in ad hoc networks under jamming using directional antennas and mobility," in *Wired/Wireless Internet Communications*, ser. Lecture Notes in Computer Science, vol. 2957. Berlin, Germany: Springer Berlin – Heidelberg, 2004, pp. 521–532.

[75] P. P. Papadimitratos, G. Mezzour, and J.-P. Hubaux, "Certificate revocation list distribution in vehicular communication systems," in *Proceedings of the Fifth ACM International Workshop on VehiculAr Inter-NETworking*, ser. VANET '08, 2008, pp. 86–87.

[76] J. Boerman and J. Bernhard, "Performance study of pattern reconfigurable antennas in MIMO communication systems," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 1, pp. 231–236, Jan. 2008.

[77] Y. Desmedt, "Unconditionally secure authentication schemes and practical and theoretical consequences," in *Proceedings of the Fifth Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '85)*, Aug. 1985, pp. 42–55.

[78] D. Puccinelli and M. Haenggi, "Multipath fading in wireless sensor networks: Measurements and interpretation," in *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, ser. IWCMC '06, 2006, pp. 1039–1044.